Alma Mater Studiorum · Università di Bologna

FACOLTÀ DI SCIENZE MATEMATICHE, FISICHE E NATURALI
Corso di Laurea Triennale in Informatica

ARCHITETTURE PER LA MOBILITÀ DEL TERMINALE

Tesi di Laurea in Architettura degli Elaboratori

Relatore:
Dott. Vittorio Ghini

Presentata da:
Michele Villani

Sessione II
Anno Accademico 2010/2011

Introduzione

Il wireless, ovvero la connettività senza fili, sta vivendo negli ultimi anni uno sviluppo sorprendente. Il suo successo è senza dubbio dovuto alla facilità con cui è possibile realizzare una Wireless Local Area Network (WLAN) e al costo molto contenuto di schede e stazioni base. La diffusione di dispositivi mobili (come cellulari, portatili o tablet) ha ulteriormente accentuato il suo sviluppo e l'introduzione di applicazioni mobili (come VoIP). Con l'allargamento dell'utenza che sfrutta questo genere di dispositivi si sono aperti molti scenari di ricerca e la mobilità su internet è diventato uno dei fenomeni caratterizzanti del nostro tempo. Cambiare la propria posizione fisica e mantenere attivi i collegamenti con la rete è quindi diventato di fondamentale importanza. Per questo motivo, l'Internet Engineering Task Force ha progettato il protocollo Mobile-IP, che, insieme con WLAN, fornisce la mobilità ad Internet.

Il progetto in cui questo lavoro di tesi si inserisce mira a confrontare le prestazioni del sistema RWMA con le prestazioni offerte dagli attuali sistemi di gestione della mobilità basati su Mobile IPv6. Cercheremo di analizzare il comportamento del **Mobile Internet Protocol version 6**[19] (MIPv6) su un nodo dotato di più intefacce wireless mentre comunica con la rete. Per fare ciò realizzeremo uno strumento di simulazione che riproduce un applicazione Voice over Internet Protocol (VoIP) in un contesto di mobilità su scala urbana.

Nel kernel del sistema operativo open-source GNU/Linux è stato realizzato il meccanismo **Robust Wireless Medium Access**[12] (RWMA) al fine di mantenere elevato il Quality of Service (QoS) per applicazioni di telefonia.

La tesi è stata divisa in più capitoli. Il primo illlustra il funzionamento delle reti wireless IEE 802.11 e dei protocolli utilizzati. Il secondo si concentrerà sulla

descrizione del RWMA e la sua implementazione su Linux. Il terzo presenterà gli strumenti di lavoro utilizzati per la realizzazione del progetto. Il quarto descriverà gli obiettivi eil funzionamento del progetto realizzato. Nel quinto capitolo analizzaremo i risultati ottenuti. Il settimo capitolo tratterà le mie conlusioni. L'ottavo descriverà gli sviluppi futuri del progetto.

Indice

Capitolo 1

Scenario

In un contesto moderno quale quello di una grande o media città, è ormai assumibile come dato di fatto che in un dato punto siano presenti una o più reti wireless. Anche se la grande maggioranza di queste sono private e presumibilmente protette, è ormai d'uso comune da parte di comuni o luoghi di ritrovo pubblici (quali alberghi, aeroporti, parchi, biblioteche, ecc), mettere a disposizione reti aperte al pubblico.

L'accesso a queste reti **WLAN**[1] è possibile grazie a qualsiasi dispositivo che sfrutti la tecnologia wireless. La più diffusa è quella basata su specifiche **IEEE 802.11**, comunemente conosciuta come **WI-FI**[2].

1.1 Infrastruttura

Una delle più comuni modalità di installazione, è chiamata **Infrastructure Basic Service Set**[3], e consiste in un insieme di **nodi** (Station e Wireless Access Point) che sono utilizzati per comunicare all'interno di una stessa BSS.

[1] Wireless Local Area Network (lett: Rete locale senza fili). Il termine solitamente indica una qualsiasi rete di dispositivi che non utilizzano dei collegamenti via cavo per connettesi alla rete.

[2] Wi-Fi, abbreviazione di Wireless Fidelity. Termine che indica dispositivi che possono collegarsi a reti locali senza fili (WLAN) basate sulle specifiche IEEE 802.11

[3] Basic Service Set (BSS) è un termine usato per descrivere una collezione di dispositivi che comunicano all'interno di WLAN (può non includere Wireless Access Point). Ne esistono di due tipi: Independent Basic Service Set e Infrastructure Basic Service Set.

Solitamente il processo attraverso il quale le station comunicano tra loro o con l'esterno è piuttosto semplice e consiste nella station che invia il suo flusso dati ad un AP, che poi si occupa di inoltrarlo al dispositivo di destinazione (all'interno della rete o all'esterno).

La tesi userà **questo tipo** di infrastruttura nelle proprie simulazioni.

Una WLAN Indipendent Basic Service Set invece, è una rete wireless (definita in modalità Ad-Hoc) che rende possibile collegare in modo indipendente più postazioni wireless tra loro senza nessun dispositivo centrale che funga da tramite. Il sistema IBSS non è adatto ad una rete numerosa e concentrata, a causa della sovrapposizione dei segnali ed i problemi che ne seguono.

1.1.1 Wireless Access Point

Un **Wireless Access Point** (abbreviato in WAP o AP) è un dispositivo, facente parte di una rete IEEE 802.11, che si presenta come un **punto di interconnessione** tra una station ed una rete, solitamente (ma non necessariamente) su cavo, quale ad esempio Ethernet.

Può essere costituito da un qualsiasi dispositivo wireless opportunamente configurato oppure da uno appositamente dedicato.

Gli AP moderni possono anche essere configurati per essere usati come:

1. Router (Instradatore): apparato che esegue l'interconnessione di reti locali multiprotocollo. Gestisce l'instradamento dei messaggi attraverso due reti.

2. Bridge (Ponte): dispositivo che interconnette due reti locali eterogenee, o ne suddivide una in più sottoreti interconnesse (viste all'esterno come una sola).

3. Client

1.1.2 Terminologia

Hotspot Nel caso gli Access Point siano pubblici, vengono definiti hotspot.

BSA La Basic Service Area è l'area teorica all'interno della quale ciascun membro di una BSS è in grado di comunicare.

Station Station (spesso abbreviato come STA) è un termine che indica qualsiasi dispositivo contenente un MAC (Medium Access Control) conforme allo standard IEEE 802.11 ed una interfaccia a livello fisico adatta a comunicare su una rete wireless. Solitamente all'interno di un contesto wireless i termini client wireless, station e nodo sono intercambiabili.

1.1.3 Schede di rete wireless

Una **Wireless Network Interface Controller** (WNIC) è una scheda di rete capace di connettersi ad una rete su mezzo radio[4].

Lavora sui livelli OSI[5] 1 e 2 ed è costruita secondo lo standard IEEE 802.11.

Alcuni parametri tipici di una scheda wireless sono:

- La banda (in Mb/s): da 2 Mbit/s a 54 Mbit/s
- La potenza di trasmissione (in dBm)
- Gli standard supportati (es: 802.11 b/g/n, ecc.)

Una scheda wireless può operare in due modalità:

1. Infrastructure
2. Ad-hoc

Nella prima modalità, la scheda ha bisogno di un AP come intermediario. Tutti i dati (di tutte le station) sono trasferiti usandolo come nodo di interconnessione. Per connettersi ad SSID[6] protette, tutte le station devono essere a conoscenza della chiave.

[4] Usa un'antenna per comunicare attraverso microonde.

[5] Open Systems Interconnection Reference Model è una descrizione astratta a livelli per l'organizzazione delle reti di comunicazioni.

[6] Il Service Set Identifier consiste in una serie di caratteri ASCII e rappresenta il nome della rete WLAN.

Nella modalità Ad Hoc invece gli AP non sono richiesti, essendo le WNIC in grado di comunicare direttamente con le altre station. Tutti i nodi devono comunque essere sullo stesso canale. L'architettura implementata nella tesi richiede che la scheda operi secondo la prima modalità.

1.2 Il protocollo IEEE 802.11

Sviluppato dal gruppo 11 dello IEEE 802 LAN/MAN Standards Committee (LMSC), il protocollo IEEE 802.11 definisce uno standard per le reti WLAN sulle frequenze 2.4, 3.6 e 5 GHz.

Tra i protocolli definiti nelle specifiche, il primo ad essere stato largamente utilizzato fu il b, seguito dal g ed infine dallo n. La tesi tratterà questi protocolli solo per quanto riguarda le sezioni pertinenti al lavoro svolto, senza soffermarsi su altri particolari comunque fondamentali dello standard, ma di interesse marginale per capire il lavoro svolto.

1.2.1 IEEE 802.11 b/g/n

Come anticipato, i protocolli b e g sono tra i più diffusi a livello civile, ed entrambi utilizzano lo spettro di frequenze sui 2,4 Ghz.

E' importante specificare che i range dei dispositivi, come quelli che verranno segnalati in seguito, possono variare in base all'ambiente in cui si trovano. Metallo, acqua e in generale ostacoli solidi riducono drasticamente la portata del segnale. Inoltre sono suscettibili a interferenze di altri apparecchi che operano sulle stesse frequenze (come forni a microonde o telefoni cordless).

Lo stesso vale per la velocità di trasferimento, i cui valori massimi sono raggiungibili solo vicino alla fonte del segnale.

IEEE 802.11 b Questo protocollo è stato ratificato nell'Ottobre del 1999. Utilizza il CSMA/CA[7] come metodo di trasmissione delle informazioni. Ha una

[7] Il Carrier Sense Multiple Access con Collision Avoidance (lett: Accesso multiplo con rilevazione di portante a eliminazione di collisione) è un protocollo di trasmissione che evita le contese, cioè i tentativi di più stazioni di accedere contemporaneamente alla rete.

capacità massima di 11Mb/s, un throughput[8] di circa 5 Mb/s e una portata massima all'esterno (in assenza di ostacoli) di circa 90-100 metri[3].

IEEE 802.11 g Nel Giugno 2003 è stata ratificato un secondo protocollo, chiamato g. È completamente compatibile con il b, ma le sue prestazioni sono nettamente maggiori. Fornisce una banda teorica di 54Mb/s, un throughput di circa 22 Mb/s e la stessa portata massima[4].

IEEE 802.11 n Lo standard è stato finalizzato l'11 Settembre 2009[5] e la sua pubblicazione è prevista per Ottobre. Compatibile con il b, il protocollo promette di essere molto più potente dei suoi predecessori, con un throughput di 144Mb/s ed una banda teorica di 600Mb/s. Come il g ed il b, la portata in spazi aperti prevista è di 90-100 metri.

1.2.2 I frame 802.11

Il protocollo 802.11 definisce come **frame** il tipo di pacchetto utilizzato sia per la trasmissione dei dati che per il controllo del canale di comunicazione. Ne esistono tre tipi:

- Dati (Data Frame): Usati per la trasmissione dei dati
- Gestione (Management Frame): Servono a scambiarsi informazioni
- Controllo (Control Frame): Gestiscono la trasmissione dei dati

Il primo tipo è quello che si occupa del trasporto vero e proprio dei dati. Gli ultimi due invece servono a creare e gestire il canale, oppure si occupano di gestire la trasmissione dei pacchetti Dati (questi tipi di frame non vengono inoltrati oltre il MAC[9] ai livelli superiori).

Per quanto riguarda la struttura di un frame, scenderemo nei particolari solo dei primi sedici bit, ovvero il **campo controllo** (presente in tutti e tre i tipi di frame). Questo è a sua volta suddiviso in undici sotto-campi (quando non specificato si assuma che siano di dimensione 1 bit):

[8] Quantità di dati trasmessa in un determinato intervallo di tempo.
[9] Media Access Control (MAC). E' un sotto-livello del livello Data Link.

1. Versione del Protocollo (2 bit) - Identificano il protocollo usato (es: 802.11g, 802.11b, ecc.)

2. Tipo (2 bit) - Specificano il sottotipo del frame: Gestione, Controllo o Dati

3. Sottotipo (4 bit) - Specificano il tipo del frame: RTS, CTS, ACK, ecc

4. Al DS - Inizializzato a 1 se il frame è diretto al sistema di distribuzione

5. Dal DS[10] - Inizializzato a 1 se il frame proviene dal sistema di distribuzione

6. Altri Frammenti (More frag) - Valorizzato con 1 solo se seguono altri frammenti appartenenti allo stesso datagram

7. Ripetizione (Retry) - Inizializzato a 1 se questo frammento è la ripetizione di un frammento precedentemente trasmesso. Aiuta l'AP nella eliminazione dei frammenti duplicati

8. Risparmio energia (Pwr mgt) - Inizializzato a 1 se al termine del frame l'interfaccia del mittente entrerà nella modalità di basso consumo . Gli AP non configurano mai questo bit

9. Altri dati (More Data) - Inizializzato a 1 se il mittente ha altri frame per il dispositivo

10. WEP – Utilizzato solo se il campo Dati è stato crittografato con l'algoritmo WEP[11]

11. Ordinati (Order) - Se è richiesto il metodo di trasmissione "strict ordering". I frame e i frammenti non sono sempre mandati in ordine perché spesso questo causa rallentamenti nella trasmissione

[10] È complementare di Al DS: uno dei due deve essere necessariamente valorizzato ed esclude l'altro.

[11] Wired Equivalent Privacy. E' un algoritmo ormai non più utilizzato per gestire la confidenzialità nelle reti wireless che implementano il protocollo IEEE 802.11.

Per quanto riguarda la comprensione del lavoro svolto, più che trattare i frame per il trasporto dei dati (DATA), ci concentreremo sui frame di gestione utilizzati per la configurazione, mantenimento e rilascio del canale di comunicazione, e quelli di controllo.

1.2.3 I frame di gestione

Tra i frame di gestione segnaliamo:

Authentication Frame Il processo di autenticazione è il meccanismo attraverso il quale un AP accetta o rigetta l'identità di una WNIC. L'interfaccia inizia il processo mandando un Frame di Autenticazione (Authentication frame) che contiene la sua identità all'AP. In una rete aperta (non criptata), la sequenza richiede solo l'invio di due frame: uno dalla scheda wireless, e uno di risposta (positiva o negativa) dall'AP.

Deauthentication Frame Un dispositivo invia un frame di de-autenticazione (deauthentication frame) a un AP quando desidera terminare una comunicazione. L'AP libera la memoria allocata e rimuove la scheda dalla tabella delle associazioni.

Association Request Frame Una interfaccia inizia il processo di associazione mandando uno di questi frame all'AP. Il frame contiene informazioni sul WNIC e l'SSID della rete a cui desidera associarsi. Se l'AP decide di accettare la richiesta, alloca le risorse per la scheda e manda un Association response frame.

Association Response Frame Contiene la risposta dell'AP ad un Association request frame. Se la risposta è positiva, trasmette anche informazioni aggiuntive (es: Association ID).

Disassociation frame Viene inviato se si vuole terminare una associazione.

Reassociation request frame Se una WNIC si allontana da un AP ed entra nel raggio di uno con un segnale più potente, invia un frame di questo tipo a quest'ultimo. Il nuovo AP coordina l'invio dei dati che possono essere ancora nel buffer del vecchio, e aspetta comunicazioni dal nodo.

Reassociation response frame Simile all'Association Response Frame, contiene la risposta alla richiesta ricevuta e le informazioni aggiuntive per la WNIC.

Beacon frame Un AP manda periodicamente dei beacon frame per annunciare la sua presenza a tutte le schede wireless nel proprio raggio di azione. Trasporta informazioni quali: timestamp, SSID, ecc. Le interfacce cercano continuamente beacon su tutti i canali radio disponibili, con lo scopo di trovare il migliore a cui associarsi.

Probe request frame Viene inviato dalla WNIC quando si richiedono informazioni riguardo gli AP disponibili nel proprio range.

Probe response frame Risposta dell'AP alla richiesta precedente, contiene tutte le informazioni necessarie a portare avanti una connessione.

1.2.4 I frame di controllo

Per ultimi abbiamo i frame di controllo, che si occupano della gestione dello scambio di dati. Ne esistono di tre tipi:

- Acknowledgement Frame (ACK)
- Request to Send Frame (RTS)
- Clear to Send Frame (CTS)

RTS e CTS Questi due tipi di frame implementano un meccanismo per ridurre le collisioni tra AP e station nascoste. Una nodo, prima di mandare i dati, invia un frame RTS come primo passo di un handshake. Una station

risponde a un frame RTS con un frame CTS dando il permesso di inviare dati. Questo tipo di gestione include un periodo di tempo in cui tutte le stazioni, tranne quella che ha richiesto il permesso, lasciano libero il canale di comunicazione.

ACK Dopo aver ricevuto un frame dati senza errori, l'AP invia un frame ACK al WNIC del mittente. Se il la scheda, a seguito di una trasmissione, non riceve un ACK entro un certo periodo di tempo, considera il frammento perso, e si appresta a inviarlo nuovamente.

1.3 Protocolli

1.3.1 Voice Over Internet Protocol

Come già anticipato, la tesi si occupa di trasmissioni Voice over IP. Il VoIP è un protocollo che permette l'invio e la ricezione di trasmissioni audio (analogico) codificate digitalmente.

L'utilizzo di una rete IP piuttosto che la rete telefonica standard, permette di comunicare con chiunque sia collegato alla stessa rete ed abbia una applicazione compatibile con la nostra a costo zero (a parte quello della connessione).

Solitamente la comunicazione consiste nello scambio di messaggi di piccole dimensioni tra due client. Chi invia, si occupa di convertire la voce in segnali digitali (una serie di bit) che verranno successivamente compressi attraverso un algoritmo al fine di ottenere pacchetti voce. Il ricevente ricostruisce, dai pacchetti, la comunicazione voce e la riproduce all'utente.

Per poter funzionare, è richiesto quindi un tipo di protocollo che permetta di trasportare la voce sotto forma di dati e ne permetta la corretta riproduzione.

1.3.2 Realtime Transport Protocol

Il protocollo RTP[12] è la scelta più comune per questo tipo di utilizzo. Trattandosi di un protocollo di livello applicazione, il mittente ed il destinatario sono client VoIP.

Il protocollo di trasporto scelto per i pacchetti RTP è solitamente UDP (su IP). La sua scelta, piuttosto che protocolli più robusti come TCP, è dovuta al fatto che UDP contiene meno informazioni sulla rilevazione di errori e sulla verifica della trasmissione, riducendo quindi il carico del pacchetto sulla rete.

Ogni piccolo frammento della comunicazione viene incapsulato in tre pacchetti (RTP, UDP e IP) dove gli header contengono le seguenti informazioni:

- RTP: Sequenza del pacchetto, timestamp e le informazioni necessarie alla corretta riproduzione del messaggio.
- Header UDP: Contiene le porte del mittente e della destinazione più semplici controlli sull'integrità del pacchetto.
- Header IP: Contiene l'indirizzo del mittente e del destinatario.

Se il protocollo RTP si occupa di garantire una buona qualità della voce, quello UDP si occupa della buona qualità della comunicazione, non imponendo al sistema di tenere un ordine stretto nella ricezione dei pacchetti.

I pacchetti vengono infatti accettati secondo l'ordine di arrivo e non quello di invio. Questo sistema permette alle applicazioni di non aspettare troppo a lungo in attesa di un pacchetto perso. Nel protocollo VoIP infatti, è spesso considerato accettabile perdere qualche pacchetto, ma non lo è avere un flusso dati in ricezione troppo lento.

[12] Realtime Transport Protocol (lett: Protocollo di trasporto Real-Time) è stato sviluppato dallo Audio-Video Transport Working Group, membro della IETF (Internet Engineering Task Force).

1.3.3 User Datagram Protocol

Lo User Datagram Protocol[13][1] (UDP) è un protocollo di livello trasporto connectionless[14] e stateless (lett: senza stato).

La comunicazione è implementata trasmettendo i pacchetti, chiamati Datagram[15], dal mittente alla destinazione senza verificare lo stato della rete ne quello del ricevente.

Infatti UDP non gestisce nessun tipo di controllo di flusso, ne garantisce l'affidabilità della connessione o l'ordine di ricezione. I pacchetti possono non arrivare o arrivare duplicati, ma nessun tipo di notifica è inviata all'utente. Ognuno dei pacchetti è completamente indipendente dall'altro.

Grazie a questi accorgimenti UDP è veloce ed efficiente. Inoltre il protocollo supporta il broadcast (l'invio di un pacchetto a tutti i nodi del network) ed il multicasting (l'invio dei pacchetti a tutti i sottoscritti ad un servizio).

Il protocollo UDP è quindi:

- Inaffidabile – Quando un pacchetto viene inviato non c'è modo di sapere se è arrivato a destinazione.
- Non ordinato – Se due messaggi sono inviati allo stesso nodo, non c'è modo di sapere l'ordine di arrivo.
- Leggero – Non ci sono meccanismi di controllo, quindi il pacchetto ha un minore carico sulla rete rispetto ad altri.
- Senza controlli – Viene controllata l'integrità dei pacchetti solo all'arrivo e solo se completi.

Gli unici servizi che UDP implementa sono il multiplexing (grazie alle porte) e il controllo dell'integrità (grazie al checksum). Ogni altro tipo di controllo deve essere implementato dall'applicazione che lo utilizza ad un livello superiore.

[13] Il protocollo è stato definito nel 1980 da David P. Reed (RFC 768).

[14] I protocolli connectionless (lett: senza connessione) sono caratterizzati dalla particolarità di non configurare una connessione end-to-end dedicata tra due nodi prima di un invio.

[15] Il termine Datagram solitamente si riferisce a pacchetti di un protocollo non affidabile.

Porte Le applicazioni UDP utilizzano dei socket datagram per gestire le connessioni. I socket legano (bind) l'applicazione a delle porte, che funzionano da punti di partenza/arrivo per i pacchetti. Una porta è una struttura identificata da un numero a 16 bit (quindi un valore che può spaziare tra 0 e 65,535) unico per ogni socket. La porta 0 è riservata.

Nel pacchetto UDP due campi sono dedicati alle porte: una del mittente e una del destinatario.

1.3.4 Internet Protocol

Internet Protocol[16][2] (IP) è il protocollo primario delle comunicazioni di rete. Definito nell'Internet Protocol Suite, ha il compito di trasportare pacchetti dal mittente alla destinazione basandosi solamente sul loro indirizzo.

Si occupa principalmente di gestire i metodi di indirizzamento e tutti i dettagli relativi al percorso (la consegna deve avvenire indipendentemente dalla struttura della rete e dal numero di sotto-reti presenti). Il livello del protocollo deve quindi conoscere la topologia di reti e delle sotto-reti per potere scegliere il giusto percorso attraverso di esse, soprattutto nel caso sorgente e destinazione siano in reti differenti.

I pacchetti del livello superiore vengono incapsulati in pacchetti chiamati datagram (come in UDP). Non è necessario stabile una connessione tra due nodi prima di comunicare (connectionless e stateless).

E' un protocollo di tipo best-effort, infatti non garantisce la consegna dei pacchetti a destinazione ne la correttezza dei dati, e condivide tutti i problemi di UDP:

- Corruzione dei dati
- Perdita dei pacchetti
- Pacchetti duplicati o non in ordine

[16]IETF RFC 791 pubblicato per la prima volta nel settembre 1981.

Esistono due versioni del protocollo IP:

- Internet Protocol Version 4 (IPv4): gli indirizzi sono di 32 bit, e per questo vi sono solo 4,3 x 10^9 indirizzi possibili. Un indirizzo consiste in una quadrupla di numeri decimali separati da un punto (es. 192.168.0.1).
- Internet Protocol Version 6 (IPv6): gli indirizzi sono di 128 bit, rappresentati da otto gruppi di 4 cifre esadecimali (es. 2001:0db8:85a3:08d3:1319:8a2e:0370:7344) offrendo un notevole spazio di indirizzamento di circa 3,4 x 10^{38} unità. Fornisce, inoltre, meccanismi di autoconfigurazione stateless[17] e statefull[18] dei nodi.

1.3.5 Mobile IP

Il Mobile IP [13] è un protocollo di gestione della mobilità che permette ad un host mobile, detto **Mobile Node,** di essere raggiungibile, indipendentemente dalla sua posizione fisica, sempre tramite lo stesso indirizzo IP e, quindi, di mantenere attive eventuali comunicazioni con altri nodi, nonostante gli spostamenti da una rete all'altra.

I problemi legati alla mobilità derivano dal fatto che un indirizzo IP identifica univocamente una connessione ad una determinata rete. Ne consegue che un'host, cambiando rete, è costretto a cambiare anche il proprio indirizzo. Poichè l'instradamento dei dati avviene sulla base del prefisso di rete del destinatario, per far si che il nodo mobile continui a ricevere pacchetti dall'host con il quale comunicava dalla rete precedente, occorre introdurre dei meccanismi che facciano da ponte tra il vecchio e il nuovo indirizzo.

[17] Stateless autoconfiguration[17]: meccanismo che consente agli host di auto-configurare le proprie interfacce di rete in base alle informazioni diffuse sulla rete dai router. Ha il grande vantaggio di non richiedere l'intervento di server centralizzati semplificando notevolmente le operazioni di configurazione ed amministrazione della rete.

[18] Statefull autoconfiguration[18]: modello di configurazione client-server, secondo il quale l'host ottiene l'indirizzo della propria interfaccia ed altri eventuali parametri da un server. É particolarmente adatta nei siti in cui siano predominanti gli aspetti legati alla sicurezza, oppure quando sulla stessa LAN convivano più reti IP distinte, nel qual caso si vuole forzare un host ad appartenere sempre e solo ad una ben determinata rete. Per le comunicazioni tra i nodi e il server viene utilizzato il protocollo DHCPv6 (ancora in fase di sviluppo).

A questo scopo il Mobile IP distingue i seguenti ruoli:

- Mobile Node (MN): il nodo mobile;
- Correspondent Node (CN): il nodo che comunica con il nodo mobile;
- Home Network (HN): la prima rete wireless alla quale si connette il nodo mobile;
- Foreign Network (FN): una rete wireless, diversa dalla home network, alla quale il nodo mobile si connette durante i suoi spostamenti;
- Home Address (HoA): l'indirizzo utilizzato dal mobile node per connettersi alla home network e che lo renderà raggiungibile indipendentemente dalla sua posizione fisica;
- Care-Of Address (CoA): l'indirizzo con il quale il nodo si connette alla foreign network;
- Home Agent (HA): un router della home network in grado di ridirigere il traffico dati destinato al nodo mobile, ovunque esso sia;
- Foreign Agent (FA, presente solo in MIPv4): il router della foreign network che ospita il mobile node; per redirigere il traffico destinato al MN, l'home agent crea un tunnel IP[19] con il foreign agent.

Esistono due versioni del protocollo Mobile IP, una per ogni tipo di protocollo IP:

- Mobile IPv4 (MIPv4)
- Mobile IPv6 (MIPv6)[19]: utilizzato in questa tesi.

[19]Tunneling IP: in generale, è una tecnica per costruire linee di collegamento virtuali point-to-point fra una coppia di nodi che sono, in realtà, separati da un numero arbitrario di reti.

In questo caso viene utilizzato per costringere un pacchetto ad essere consegnato in un luogo particolare, anche se la sua destinazione originale potrebbe portare il pacchetto in un luogo diverso. Per creare l'illusione che il pacchetto viaggi in un tunnel, viene incapsulato in un nuovo pacchetto prima di essere instradato, e decapsulato una volta giunto a destinazione. In questo modo, solo i nodi collegati dal tunnel sono in grado di leggere il pacchetto originale.

Capitolo 2

RWMA

Per comprendere la simulazione è necessario soffermarci anche sulla soluzione proposta dal meccanismo Robust Wireless Medium Access (RWMA)[12] per garantire un livello di prestazioni tale da permettere a dispositivi mobili, muniti di più schede di rete wireless, di effettuare comunicazioni VoIP.

2.1 Architettura RWMA

L'architettura RWMA consente ad un end-sytem wireless mobile di avere comunicazioni VoIP con un altro end-system, fisso o mobile, situato in una diversa rete, fornendo supporto per la QoS[1].

[1]Il temine Quality of Service (lett: Qualità del servizio) si riferisce a protocolli che si occupano di garantire determinati livelli di performance nelle trasmissioni dati.

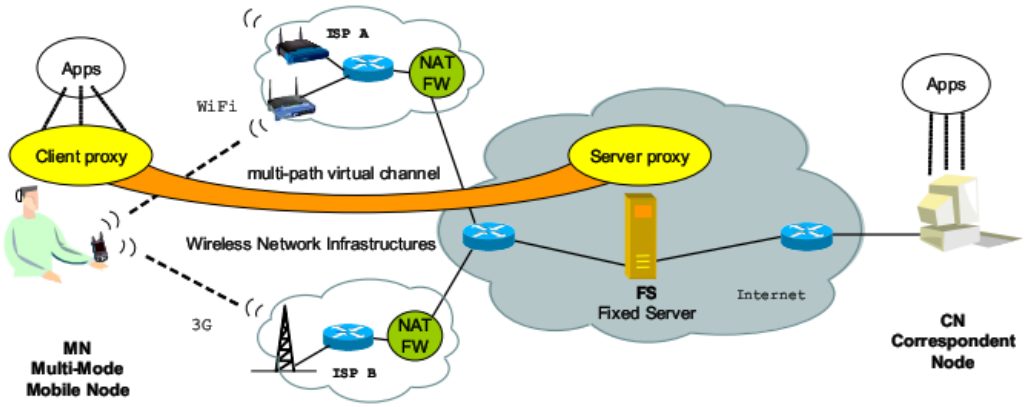

Figura 2.1: Possibile scenario di utilizzo dell'architettura RWMA

RWMA è strutturata in modo da garantire una grande interattività e una bassa perdita di pacchetti nel contesto di applicazioni VoIP. Il sistema si occupa di configurare e mantenere collegamenti wireless distinti con più AP, usando uno di questi alla volta per inoltrare il proprio traffico dati al destinatario. Inoltre durante la trasmissione, si assicura che la connessione non violi i requisiti QoS prefissi, e nel caso di tale violazione, sospende l'utilizzo del collegamento per continuare la propria trasmissione su un altro.

Questo meccanismo lavora in modo trasparente alle applicazioni VoIP che ne fanno uso, e può essere utilizzato su qualsiasi station che sia equipaggiata con più di una interfaccia wireless.

2.1.1 Modello QoS

Il livello di QoS garantito da RWMA prevede:

1. Perdita dei pacchetti inferiore al 3% di quelli trasmessi
2. Tempo di trasmissione inferiore ai 150 ms

Per godere di tale benefici, la station deve avere a propria disposizione interfacce wireless multiple, e deve essere in presenza di più reti wireless a cui connettersi.

Il sistema è implementato su più livelli OSI ed ha come fine ultimo quello di realizzare un meccanismo di controllo sulla correttezza della trasmissione su comunicazioni di tipo UDP. Questo permette al livello applicazione, se necessario, di portare avanti delle procedure di recupero, e decidere quale interfaccia wireless, tra quelle disponibili, debba essere utilizzata per le trasmissioni successive.

2.1.2 Infrastruttura

Questa architettura prevede due applicazioni, una su una station ed una su un server, che fungano da proxy per le applicazioni VoIP.

L'implementazione sulla station si occupa di riceve dei pacchetti RTP dall'applicazione, di incapsularli in datagram UDP e infine trasmetterli (e se necessario ritrasmetterli) alla destinazione.

Quella nel server invece riceve i datagram, li riordina, e li passa all'applicazione per la riproduzione. Prerequisito è che il server che riceve il messaggio, deve essere a conoscenza di quale interfaccia è stata usata per inviare pacchetto (al fine di rispondere correttamente).

Il sistema implementato nella station è formato da tre componenti:

1. Transmission Error Detector (TED)
2. UDP Load Balancer (ULB)
3. Monitor

2.1.2.1 Transmission Error Detector

Il TED è la componente più importate del meccanismo RWMA ed opera al livello MAC del protocollo IEEE 802.11b/g/n/e[3, 4, 8, 6].

Si occupa di controllare se ogni singolo datagram UDP è stato ricevuto con successo, attraverso un collegamento configurato e attivo, dallo Access Point, oppure se è stato scartato dal livello MAC.

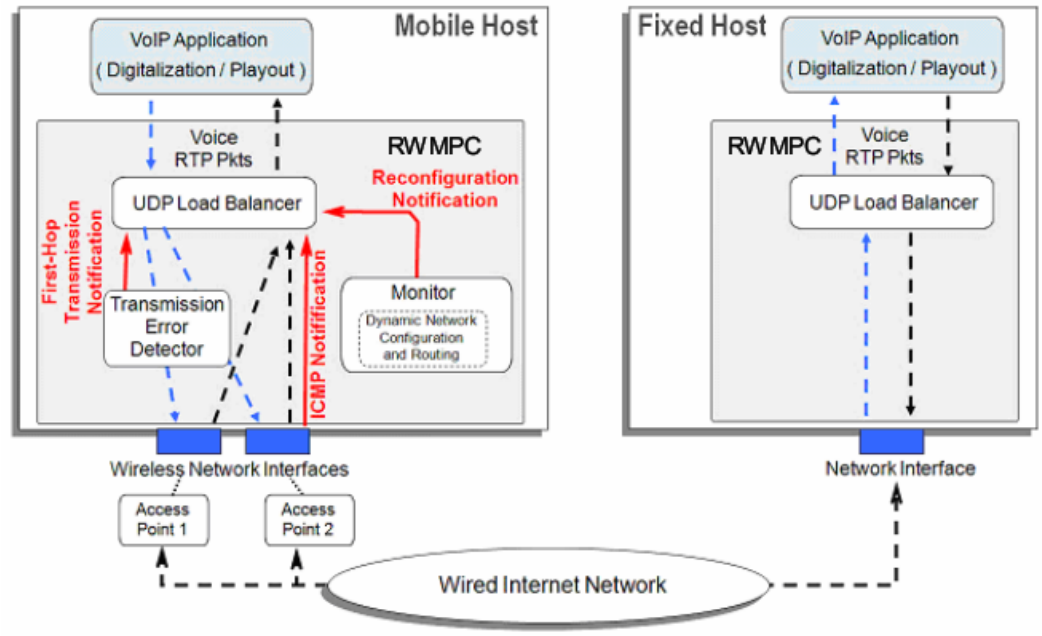

Figura 2.2: Architettura del sistema RWMA

Inoltre comunica al Load Balancer informazioni concernenti lo stato della trasmissione dei datagram UDP, attraverso il "First-hop Transmission Notification" (lett: Notifica della trasmissione del primo hop).

Ack del primo hop Utilizzando UDP su IP come protocollo di trasporto, non abbiamo meccanismi di controllo sulla effettiva ricezione del nostro pacchetto alla destinazione. L'unico controllo su cui possiamo fare affidamento è la segnalazione sull'effettiva ricezione del frame da parte del primo hop (solitamente un Wireless Access Point[2]).

Questo permette di implementare un semplice meccanismo per il controllo del flusso della trasmissione a basso livello per pacchetti UDP, protocollo che di per se non godrebbe di nessun tipo di controllo simile. È gestito fino al primo hop della comunicazione, da qui il nome della notifica.

[2]Nella maggior parte dei casi i dispositivi WAP sono connessi direttamente alla rete cablata, il che diminuisce notevolmente la possibilità di perdita dei pacchetti.

Il meccanismo, una volta implementato, permette all'ULB di capire se la trasmissione di un frammento al Wireless Access Point è andata a buon fine.

La notifica avviene attraverso l'uso di un messaggio apposito che viene inserito nella coda dei messaggi di errore del socket utilizzato per inviare il messaggio.

2.1.2.2 UDP Load Balancer

La componente ULB è situata a livello applicazione. Quella nel mittente, si occupa di ricevere i pacchetti RTP generati dall'applicazione VoIP, incapsularli in datagram UDP ed inviarli alla destinazione. Riceve notifiche di errore da più fonti, alla luce delle quali decide se il pacchetto deve essere ritrasmesso o scartato.

Inoltre, sempre basandosi su queste informazioni, decide quale interfaccia, tra quelle disponibili, utilizzare per mandare il prossimo datagram.

L'ULB riceve tre tipi di notifiche:

- Reconfiguration Notification (fonte: Monitor)
- First-hop Transmission Notification (fonte: TED)
- ICMP error

Gli ultimi due tipi di errore vengono ricevuti attraverso la system call recvmsg, settando la flag MSG_ERRQUEUE.

L'ULB sul lato del ricevente è più semplice. Ha una sola interfaccia disponibile e si assume che il pacchetto non venga mai scartato nel primo hop, essendo connesso con una interfaccia wired (quindi non ha bisogno di ricevere notifiche).

Si occupa di registrare l'indirizzo IP dell'ultima interfaccia wireless da cui ha ricevuto un UDP datagram per poter rispondere con successo.

Reconfiguration Notification Notifica che una data interfaccia è stata configurata o disabilitata. Ognuna delle interfacce presenti sulla station viene configurata dal Monitor, e per ognuna di esse l'ULB genera un socket UDP e invoca la bind su di esso, per inviare e ricevere messaggi. Quando una

interfaccia viene disabilitata, l'ULB chiude il socket corrispondente e considera persi i messaggi inviati su di esso e per cui non è stato ancora ricevuto nessun messaggio di tipo "First-hop Transmission Notification".

First-hop Transmission Notification Notifica che un particolare pacchetto è stato ricevuto con successo dall'AP o è stato scartato.

Errore ICMP Notifica che un datagram UDP è stato perso lungo il percorso tra mittente e destinatario.

2.1.2.3 Monitor

Questa componente si occupa di monitorare e configurare le interfacce wireless della station e le routing tables[3].

All'avvenuta configurazione di una WNIC si occupa di notificare l'ULB, specificando quale scheda di rete è stata correttamente configurata ed è pronta per l'utilizzo oppure quale è stata disabilitata. Questa notifica prende il nome di "Reconfiguration Notification".

2.2 Implementazione di RWMA su Linux

Come anticipato, l'architettura RWMA è stata implementata con successo su un sistema Linux (Gentoo) con un kernel versione 2.6.27.4.

2.2.1 Implementazione del Monitor

Il monitor è stato implementato come un'applicazione separata che comunica con il kernel linux attraverso l'uso di socket Netlink[4].

Usa come base un'applicazione open-source chiamata **wpa_supplicant**, che si occupa di gestire gran parte delle operazioni associate alla gestione di

[3]La Routing Table (lett: tabella di instradamento) è una tabella usata per instradare un datagram al suo prossimo hop.

[4]Netlink è un meccanismo basato su socket per la comunicazione tra il kernel e i processi user space, o tra processi user space (come i socket unix). I socket Netlink possono comunicare solo all'interno dello stesso host, essendo il loro indirizzamento basato su PID.

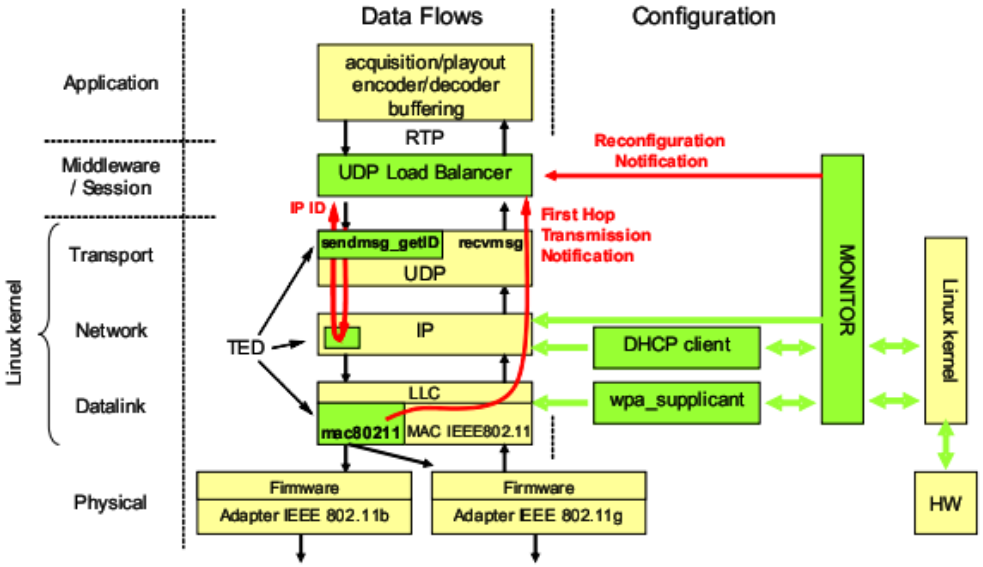

Figura 2.3: Implementazione RWMA su Linux

schede wireless. Ad esempio, attiva la scansione dei canali radio alla ricerca di Access Point e di connessioni wireless di cui siano a conoscenza le informazioni necessarie all'autenticazione. Nel caso siano presenti più AP, è in grado di scegliere quello con il segnale radio migliore.

Una volta associati con un AP, il kernel informa il monitor che avvia un client DHCP[5]. Quando la configurazione è terminata con successo, il monitor setta una nuova regola ed una nuova rotta sulla routing table, per permettere un routing dinamico attraverso quella scheda. Questo fa sì che i datagram IP possano essere indirizzati basandosi sull'indirizzo del mittente e non su quello di destinazione.

Inoltre, una volta che la scheda è stata configurata, questo viene segnalato all'ULB che può iniziare ad usarla.

È stata anche effettuata una piccola modifica affinché più schede wireless non si connettano con lo stesso AP, e quindi il livello applicazione abbia a disposizione percorsi differenziati per raggiungere l'end system.

[5]Dynamic Host Configuration Protocol (lett: Protocollo di configurazione dinamica degli indirizzi) è un protocollo che permette ai dispositivi di rete di ricevere la configurazione IP necessaria per poter operare su una rete basata su tale protocollo.

Quando una WNIC non riceve più beacon da un AP al quale è associato, il monitor disabilitata la scheda, porta avanti le operazioni di dissociazione e cancella le informazioni che la riguardano dalla routing table. Questa interfaccia sarà inutilizzabile finché non verrà effettuata una riconfigurazione.

2.2.2 Implementazione del Load Balancer

Questa componente è stata implementata a livello middleware.

Ricevute le notifiche dal Monitor sulle schede configurate, si occupa di creare un UDP socket per ognuna di loro e di invocare la bind(). Inoltre, utilizzando le rotte create dal Monitor, tutti i datagram IP su un dato socket vengono indirizzati attraverso la stessa scheda wireless (di conseguenza tutti i datagram che si riferiscono ad un dato socket hanno lo stesso indirizzo IP come mittente).

In altre parole, l'ULB può selezionare una scheda wireless da usare per mandare il datagram utilizzando l'apposito socket UDP.

Quando l'ULB invia un datagram UDP all'end system di destinazione, deve essere informato se l'AP lo ha ricevuto o lo ha scartato. Per ottenere questo, utilizza due metodi:

- Una nuova system call chiamata sendmsg_getID.
- Un sistema di notifica

La system call sendmsg_getID, inclusa nei moduli UDP e IP del kernel, estende il comportamento di una system call esistente: la sendmsg()[6].

La funzione, oltre ad inviare il pacchetto come la chiamata originale, ritorna all'applicazione un identificativo univoco (intero) che identifica il datagram IP che viene incapsulato in quello UDP (nient'altro che il campo indentification del datagram IP, temporaneamente univoco).

L'ULB mantiene inoltre un elenco di questi pacchetti, aspettando una notifica di successo o fallimento dell'invio da parte della componente TED nel livello MAC, trasportata su un pacchetto di tipo IP_NOTIFY. Anche questa notifica contiene l'id del datagram in questione.

[6]Responsabile di trasmettere un datagram UDP a una destinazione dato un socket UDP.

Questi messaggi possono essere letti invocando la recvmsg con il flag MSG_ERRQUEUE.

2.2.3 Implementazione del TED

L'implementazione del TED è suddivisa in due parti: la prima al livello trasporto e rete, la seconda al sub-livello MAC del livello datalink.

Il firmware si occupa di portare avanti una trasmissione asincrona all'AP e ritornare il risultato di questa al livello MAC. Il TED riceve questa notifica e, se il frame contiene un datagram UDP, estrae il risultato (estrae le porte UDP del mittente dal primo frammento di ogni datagram), trova il socket che è stato usato per inviarlo, e lo informa mandando un messaggio particolare nella sua coda dei messaggi di errore.

Per verificare se un pacchetto ha richiesto il servizio RWMA, viene controllato se il socket su cui è memorizzato il pacchetto è stato configurato correttamente.

Ogni socket ha una coda interna dei messaggi nei quali i messaggi ICMP vengono memorizzati. L'applicazione può leggere da questa coda configurando appositamente il socket (settando il flag IP_RECVERR attraverso la system call setsockopt). In questa implementazione è stato introdotto un nuovo tipo di messaggio chiamato IP_NOTIFY. Questo messaggio contiene:

1. Id del datagram IP
2. Il risultato della trasmissione
3. La lunghezza del frammento
4. Il valore del campo morefragment
5. Il campo offset del frammento

Capitolo 3

Framework

In questo capitolo verranno introdotti i concetti fondamentali necessari a comprendere il lavoro svolto nella tesi e le scelte implementative che verranno discusse nei capitoli successivi. Verrà spiegato cosa sono e come funzionano i framework OMNeT++[9] e xMIPv6[16], su cui si basa l'intera implementazione.

3.1 OMNeT++

OMNeT++ è un framework modulare per lo sviluppo di simulazioni ad eventi discreti. Non è propriamente un simulatore, ma piuttosto una infrastruttura che fornisce i mezzi per scrivere delle simulazioni.

Il campo in cui sta subendo lo sviluppo maggiore è sicuramente quello delle reti per la comunicazione. Ad ogni modo, la sua architettura generica e flessibile gli permette di essere usato con successo anche in altri ambiti, tra i quali:

- Modellazione di protocolli generici
- Modellazione di sistemi multiprocessore o distribuiti
- Validazione di architetture hardware
- Valutazione delle performance di sistemi software

Più in generale, OMNeT++ può essere usato in qualsiasi contesto dove un approccio ad eventi discreti è applicabile, e le cui entità possono essere mappate in moduli che comunicano attraverso lo scambio di messaggi[1]. Il suo diffusissimo uso sia nella comunità scientifica che in quella industriale, lo rende un'ottima ed affidabile scelta per l'implementazione e lo studio del meccanismo RWMA.

La versione di OMNeT++ a cui si fa riferimento è la 4.1 (stabile) rilasciata il 14 Giugno 2010[10].

3.1.1 Moduli

Le componenti fondamentali del framework sono delle classi riusabili chiamate **moduli**, scritte in C++ usando le librerie di OMNeT++.

I moduli sono strutturati in maniera gerarchica, fino a modellare una struttura il cui numero massimo di livelli non è definito. Quelli al livello più basso vengono chiamati **simple modules** (moduli semplici), e rappresentano delle entità e/o dei comportamenti. Il modulo di massimo livello invece viene chiamato **system module** (modello di sistema) e racchiude tutto il sistema nel suo complesso.

I moduli semplici vengono assemblati in componenti più grandi e complessi chiamati **compound modules** (moduli composti) o in **modelli** (chiamati anche network), attraverso l'uso di un linguaggio di alto livello chiamato **NED**. Un modello è di per sé un modulo composto.

I moduli semplici possono avere parametri che vengono principalmente utilizzati per passare dati di configurazione ai moduli semplici, o per parametrizzarne il comportamento (possono essere stringhe, numeri o valori booleani). All'interno di un modulo composto invece i parametri possono definire il numero di sotto moduli, gate o connessioni interne.

I parametri possono essere assegnati nei file NED, nei file di configurazione (con estensione ini) o possono essere chiesti interattivamente all'utente al lancio della simulazione. Possono riferirsi ad altri parametri o essere il frutto di calcoli su di essi.

[1] Tutto OMNeT++ è costruito sul meccanismo del Message Passing.

3.1.2 Gate

Solitamente i moduli semplici inviano e ricevono messaggi attraverso dei gate (associabili al concetto di porte), i quali sono in tutto e per tutto le interfacce di input ed output di un modulo.

Un gate può essere collegato con una **connessione** creata all'interno di uno stesso livello gerarchico, oppure può collegare un modulo semplice con uno composto. Le uniche connessioni vietate sono quelle tra diversi livelli gerarchici, perché andrebbero a minare la riusabilità del modello stesso. Data la struttura gerarchica, i messaggi viaggiano attraverso una catena di connessioni, per partire ed arrivare in un modulo semplice attraverso dei gate.

Una connessione che collega due moduli semplici viene anche definita **route** (rotta) o **link**.

Le connessioni possiedono tre parametri (tutti opzionali):

- Propagation delay: lasso di tempo di cui il pacchetto viene rallentato nel mezzo prima di essere consegnato.
- Bit error rate: probabilità che un bit venga trasmesso in maniera non corretta.
- Data rate (bit/s): viene usata per calcolare il tempo di trasmissione di un pacchetto.

3.1.3 Messaggi

All'interno della simulazione, i messaggi possono rappresentare frame, pacchetti di un network, job o qualsiasi altro tipo di struttura arbitraria. Hanno attributi standard (quali il timestamp) e possono arrivare da un altro modulo oppure dal modulo stesso, nel qual caso vengono definiti **self-message** (lett: auto-messaggi) il cui invio è gestito con dei timer.

Il "tempo locale di simulazione" di un modulo è strettamente legato ai messaggi. Questo infatti aumenta all'arrivo nel modulo di uno di essi.

Gli header dei pacchetti sono descritti nei **Message Definition File** (file semplici con estensione msg), scritti in una sintassi simile a C. Questi file ven-

dono tradotti in classi ed header C++ dal tool OMNeT++ **opp_msgc**. Per esempio, un nuovo pacchetto chiamato pacchetto.msg, dato in input al programma opp_msgc genera due file C++ chiamati rispettivamente: pacchetto_m.h e pacchetto_m.cc. Per poter creare ed utilizzare i pacchetti qui definiti all'interno della propria implementazione, è necessario includere l'header file del pacchetto nella propria classe.

Le classi generate in questo modo sono sottoclassi di cMessage (libreria OMNeT++).

3.1.4 Comunicazione tra livelli di protocollo

In OMNeT++ quando un protocollo di un livello superiore vuole mandare il pacchetto su un protocollo di livello inferiore, manda l'oggetto attraverso la connessione che li lega al modulo sottostante, che poi si occuperà di incapsularlo ed inoltrarlo.

Il processo inverso avviene quando un modulo di un livello inferiore riceve un pacchetto. In questo caso il modulo si occupa di mandarlo al livello superiore dopo averlo decapsulato.

Control Info A volte è necessario veicolare informazioni aggiuntive insieme al pacchetto. Questo tipo di informazioni viaggiano in un oggetto chiamato **control info**, che viene collegato al messaggio attraverso la chiamata setControlInfo() del pacchetto, e contiene informazioni ausiliarie che sono necessarie al livello a cui è diretto, ma che non sono da inoltrare ad altri moduli. Gli oggetti control info possono ovviamente essere definiti dall'utente, e sono sottoclassi di cObject (libreria OMNeT++).

3.1.5 Linguaggio NED

Il termine NED fa riferimento a **Network Description** (lett. Descrizione del network), ed è un linguaggio di alto livello usato dall'utente per descrivere la struttura di un modello.

Ha una struttura gerarchica e flessibile (a package) simile a Java, per ridurre il rischio di name clashes tra moduli differenti. Un esempio è il NEDPATH

(simile al CLASSPATH di Java), che rende più facile specificare dipendenze tra moduli.

Altra particolarità di questo linguaggio è la sua struttura ad albero, del tutto equivalente ad XML[2]. Un file con estensione ned può quindi essere convertito in XML (o viceversa) senza perdita di dati, inclusi i commenti.

Il NED permette di creare moduli semplici, e successivamente connetterli ed assemblarli in moduli composti, di ottenere sottoclassi, aggiungere parametri, gate e nuovi sotto-moduli e di assegnare parametri esistenti a valori fissi o casuali.

Quando vengono modificati file ned non è necessaria una ricompilazione, essendo tutto gestito a run-time.

3.1.6 File .ini

Questo tipo di file contiene la configurazione ed i dati di input per le simulazioni. I dati al suo interno sono raggruppati in sezioni il cui nome, racchiuso tra parentesi quadre (e dopo la parola chiave Config), indica l'identificatore univoco della simulazione. L'uso di wildcard e dell'ereditarietà tra le simulazioni permette una configurazione veloce di un gran numero di moduli contemporaneamente.

In questi file si fa riferimento ai parametri dei moduli attraverso il loro path completo o al loro "nome gerarchico". Quest'ultimo consiste in una lista di nomi di moduli separati dal "." (dal modulo di massimo livello al modulo contenente il parametro).

I parametri vengono assegnati attraverso l'operatore "=", e la loro risoluzione avviene nel seguente modo:

1. Se il parametro è già assegnato nel NED, questo non può essere sovrascritto

2. Se dopo l'operatore di assegnamento è presente un valore, questo viene assegnato al parametro

[2]XML (Extensible Markup Language (XML) è un formato di codifica per il testo semplice e flessibile, derivato da SGML (ISO 8879).

3. Se dopo l'operatore di assegnamento è presente l'identificatore "default", viene assegnato il valore di default per il tipo del parametro

4. Se dopo l'operatore di assegnamento è presente l'identificatore "ask", il valore da assegnare viene chiesto in modo interattivo all'utente

5. Se il parametro non viene assegnato ma ha un valore di default definito, questo viene assegnato

6. Se non si è in presenza di nessuno dei casi sopra citati, il parametro viene dichiarato "non assegnato" e verrà gestito a seconda della politica dell'interfaccia utilizzata

3.1.7 OMNEST

OMNEST è la versione commerciale di OMNeT++. Il framework infatti è libero solo per un uso accademico senza scopo di lucro. Per utilizzi commerciali è necessario acquistare una licenza OMNEST da Simulcraft Inc.

3.2 xMIPv6

Il framework xMIPv6 è costruito al di sopra OMNeT++, e si basa sullo stesso concetto: moduli che comunicano attraverso l'invio di messaggi. È interamente open-source, e viene usato principalmente per la simulazione di comunicazioni di rete.

Implementa i più diffusi protocolli di rete come UDP, TCP, SCTP, IPv6, MIPv6, Ethernet, IEEE 802.11 e altri ancora.

3.2.1 Protocolli

I protocolli sono rappresentati da moduli semplici la cui implementazione è contenuta in una classe che solitamente porta il loro nome. Le interfacce di rete (Ethernet, 802.11, ecc), invece sono solitamente dei moduli composti.

Questi moduli possono essere liberamente ricombinati per formare station o altri dispositivi a seconda delle necessità. Diversi tipi di host, router, switch

ed access point sono già presenti all'interno del codice di xMIPv6 nella cartella "nodes", ma ovviamente ne possono essere creati di nuovi su misura per il proprio scenario.

Non tutti i moduli semplici però implementano protocolli. Ci sono moduli che contengono informazioni (es: RoutingTable6), gestiscono la comunicazione (es: NotificationBoard), l'auto-configurazione di un network (es: FlatNetworkConfigurator6), il movimento dei nodi (es: TurtleMobility), oppure gestiscono operazioni sui canali radio nelle comunicazioni wireless (es: ChannelControl).

3.2.2 Moduli comuni

Ci sono moduli che grazie all'importantissimo ruolo ricoperto, sono quasi indispensabili all'interno di host, router ed altri dispositivi di rete. Tra questi segnaliamo:

- InterfaceTable: questo modulo tiene memoria della tabella delle interfacce (eth0, wlan0, ecc) negli host. Non manda ne riceve messaggi ed è accessibile dagli altri moduli attraverso una semplice chiamata C++. Le schede di rete si registrano (inseriscono nella tabella) dinamicamente implementandone l'interfaccia.

- RoutingTable6: questo modulo gestisce le routing table per IPv6 e viene acceduto dai moduli che sono interessati a lavorare con le rotte dei pacchetti (soprattutto IP). Ci sono funzioni per richiedere, aggiungere, cancellare, e trovare le rotte migliori per un dato indirizzo IP.

- NotificationBoard: permette ai moduli di comunicare secondo un modello publish-subscribe. Lavorando in questa modalità, quando un modulo genera un evento, questo viene notificato a tutti i moduli che lo hanno sottoscritto. Nessuno scambio di messaggi è richiesto.

- ChannelControl: necessario nelle simulazioni wireless, tiene traccia dei nodi e della loro posizione.

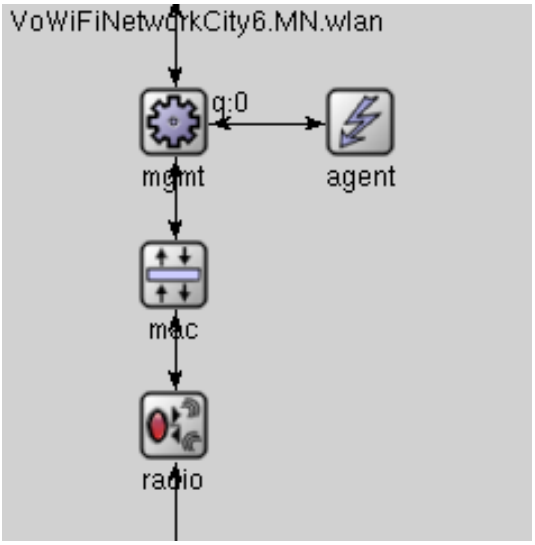

Figura 3.1: Architettura di una WNIC in INET

3.2.3 Architettura di una NIC IEEE 802.11

Una interfaccia NIC (Ieee80211) nel framework xMIPv6 consiste nei seguenti quattro moduli composti tra loro (in ordine top-down):

1. Agent
2. Management
3. MAC
4. Livello fisico (radio)

L'agent L'agent è il modulo che si occupa di gestire il comportamento del livello a lui annesso (management). Si occupa di ordinare (attraverso messaggi command) a quest'ultimo di condurre operazioni quali la scansione dei canali radio, l'autenticazione o l'associazione con un Access Point. Il livello management si limita ad eseguire questi comandi per poi riportare il risultato all'agent. Modificando o rimpiazzando un agent, si può modificare il comportamento stesso di uno STA ed implementare algoritmi o strategie necessari alla propria simulazione.

Il manager Il manager si occupa di incapsulare e decapsulare i messaggi per/dal MAC, e scambiare frame di gestione con altre station o AP. I frame Probe Request/Response, Authentication, Association Request/Response ecc, sono creati ed interpretati dal manager ma trasmessi e ricevuti attraverso il MAC. Durante la scansione è il manager che cambia periodicamente canale per raccogliere informazioni e ricevere beacon e probe response. Come l'agent, ha differenti implementazioni a seconda del suo ruolo.

Il livello MAC Il livello MAC si occupa della trasmissione dei frame secondo il protocollo CSMA/CA[3]. Riceve dati e management frame dai livelli alti, e li trasmette.

Il livello fisico Il livello fisico si occupa della trasmissione e ricezione dei frame. Modella le caratteristiche del canale radio e determina se un frame è stato ricevuto o no correttamente (ad esempio nel caso subisca errori a causa del basso potere del segnale o interferenze nel canale radio). I frame ricevuti correttamente sono passati al MAC.

3.2.4 Simulazioni

Le simulazioni in OMNeT++/xMIPv6 possono essere eseguite in diverse modalità. Mentre l'interfaccia grafica (default) è comoda per dimostrazioni ed il debugging, la versione da linea di comando è sicuramente quella più adatta per esecuzioni batch.

Durante una simulazione tutti i campi di una classe (se appositamente inizializzati) possono essere controllati, navigati e modificati.

Per eseguirne una, bisogna recarsi nella cartella dove è situato il file di configurazione (quello con estensione ini) e lanciare l'eseguibile di xMIPv6 con il file come parametro.

- /PATH_TO_XMIPV6/run_xmipv6 <*fileDiConfigurazione.ini*>

[3] Carrier Sense Multiple Access (CSMA) è un protocollo MAC probabilistico nel quale un nodo verifica l'assenza di altro traffico prima di trasmettere su un canale condiviso.

Questo tipo di esecuzione lancia una interfaccia grafica. Le simulazioni definite nel file (nel caso ce ne siano più di una) possono essere scelte da un comodo menù a tendina nella finestra della simulazione. Mentre nel caso si voglia lanciare direttamente una simulazione specifica all'interno del file, si utilizza la chiamata:

- /PATH_TO_XMIPV6/run_xmipv6 -c <*nomeDellaSimulazione fileDiConfigurazione.ini*>

Capitolo 4

Progettazione

4.1 Obiettivi

Il progetto in cui questo lavoro di tesi si inserisce mira a confrontare le prestazioni del sistema RWMA precedentemente descritto con le prestazioni offerte dagli attuali sistemi di gestione della mobilità basati su Mobile IPv6 ed anche sulle futute estensioni di Mobile IPv6. A questo scopo è gia' stato costruito un simulatore dell'architettura RWMA operante in un contesto cittadino. In questo lavoro di tesi, invece, da un lato si vuole costruire un simulatore che realizzi l'architettura Mobile IPv6 nello stesso scenario cittadino, dall'altro lato si vuole aggiungere a questo simulatore le caratteristiche necessarie ad un futuro sviluppo del simulatore. Questo sviluppo futuro del simulatore dovra' realizzare i protocolli delle estensioni di Mobile IPv6 note col nome di "Multiple Care of Address"[22] che attualmente non sono ancora state stabilizzate in uno standard. Queste estensioni di MIPv6 mirano a gestire contemporaneamente piu' interfacce di rete in uno stesso nodo mobile. Quindi, in questo lavoro di tesi, si vuole costruire un simulatore che realizzi l'architettura Mobile IPv6 nello scenario cittadino e che disponga di nodi mobili dotati di piu' interfacce wireless per poter in un futuro realizzare le politiche per il "Multiple Care of Address"

Per ottenere un'analisi approfondita del comportamento del protocollo MobileIPv6 è stato scelto di utilizzare una sua versione simulata. Per lo sviluppo

dello scenario ci siamo quindi basati sulla simulazione, realizzata da Silvia Ripa[14], "VoWiFiNetworkCity6" ancora in fase di sviluppo, che fornisce una rete con nodo mobile MIPv6 dotato di una sola scheda di rete.

La realizzazione della simulazione è ancora in fase di sviluppo, ci limiteremo quindi a creare i presupposti per un approfondimento futuro.

Il lavoro è stato diviso in due fasi:

- progettare e realizzare di un modello simulativo che riproduca l'approccio basato sul protocollo MIPv6 dotato di più schede di rete wireless;
- realizzare dei test per verificare il comportamento dello scenario simulato.

Il progetto si basa sulla simulazione "VoWiFiNetworckCity6" che sfrutta una rete IPv6 su scala urbana. La simulazione riproduce un approccio che consente ad un nodo mobile (basato su MIPv6) di trasmettere messaggi VoIP attraverso la rete.

Inizieremo descrivendo la struttura dello scenario "VoWIFiNetworkCity6" realizzata da Silvia Ripa e poi le scelte implementative realizzate per raggiungere l'obbiettivo della tesi.

4.2 Infrastruttura rete

La rete Internet reale è un'interconnessione di reti LAN, MAN e WAN[1] in cui host di reti diverse sono in grado di comunicare grazie alla presenza di nodi (router) incaricati di instradare i pacchetti tra le diverse reti. E' quindi indispensabile, per riprodurre una comunicazione VoIP realistica, ricostruire nella simulazione una rete, seppure in miniatura, che si avvicini al funzionamento di Internet.

L'intera rete, in VoWiFiNetworkCity6, è formata da un insieme di sottoreti LAN e WLAN interconnesse tra loro e raggruppate in due reti collegate da due backbone di diversa latenza. In particolare uno dei due backbone presenta una

[1]Local Area Network, Metropolitan Area Network e Wide Area Network sono le vari tipologie di reti presenti in una internet.

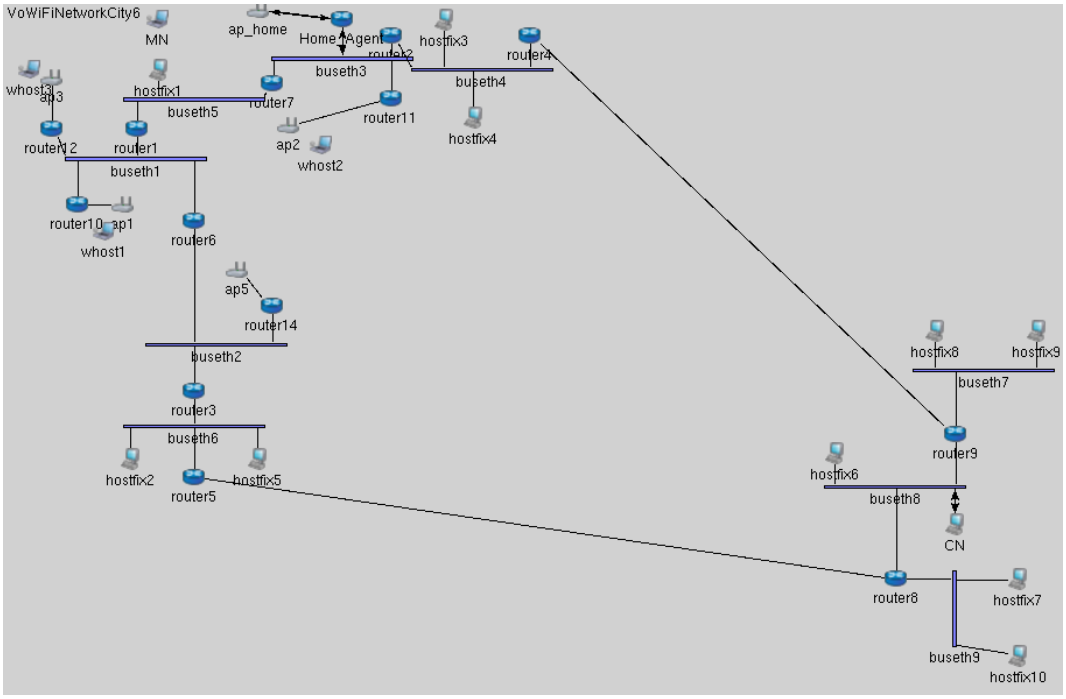

Figura 4.1: Struttura della rete implementata in VoWiFiNetworkCity6

latenza inaccettabile per una conversazione VoIP. I nodi e i router utilizzano, a livello di rete[2], il protocollo IPv6.

Tipologia delle subnet La tecnologia utilizzata per le reti wireless è la WiFi (IEEE 802.11) e ciascuna è costituita da un access point (AP) e un nodo wireless. Le reti su cavo (wired), invece, utilizzano la tecnologia ethernet (IEEE 802.3) e sono costituite da un bus e da un insieme di nodi collegati ad esso.

Traffico di background wired Viene generato da tutti gli host presenti nella rete, che comunicano tra di loro trasmettendo pacchetti ad intervalli temporali prefissati. In particolare, gli host nella rete in cui si muove il nodo mobile comunicano con quelli nella rete in cui si trova il correspondent node[3].

[2]Livello 3 dello standard ISO/OSI

[3]Correspondent Node (CN): nel protocollo MIPv6 rappresenta il nodo che comunica con il nodo mobile.

Traffico di background wireless Viene generato da nodi wireless posizionati in prossimità degli access point. Le stations comunicano solo tra di loro, ed in particolare una di queste genera un flusso continuo di dati in modo da provocare interferenze al nodo mobile che si sposta in quella particolare sottorete.

Backbone I due backbone implementati sono rappresentati da due connessioni di tipo ethernet (IEEE 802.3) fra i router di bordo delle due reti principali. Per ciascun backbone è stata definita una propria latenza interna. In particolare ad uno è stato attributo una latenza inferiore a 150ms, accettabile per le comunicazioni VoIP, mentre all'altro, una elevata di 200ms.

4.3 Comunicazioni VoIP

Per simulare una comunicazione VoIP, il nodo mobile e il correspondent node scambiano, con una frequenza di 40 ms, pacchetti UDP di 100 Byte.

4.4 Configurazione degli indirizzi

Per gestire la configurazione degli indirizzi dei nodi è stato utilizzato il modello **stateless autoconfiguration**[17], che consente agli host di auto-configurare le proprie interfacce in base alle informazioni diffuse sulla rete dai router. Il processo si compone di alcuni passi[20]:

1. Viene generato automaticamente un indirizzo di tipo link-local (a partire dal MAC address associato all'interfaccia) all'attivazione di un'interfaccia.

2. Si inizia una procedura di Duplicate Address Detection (DAD), per assicurarsi che il nuovo indirizzo sia unico all'interno del link. Se l'indirizzo è già utilizzato da un altro host sul link, viene generato un errore ed occorre intervenire mediante configurazione manuale; viceversa se la procedura conferma l'unicità dell'indirizzo, questo viene assegnato all'interfaccia interessata. In questo modo il nodo possiede un indirizzo locale utilizzabile

per comunicare con tutti gli altri nodi on-link, ossia tutti gli host a cui è direttamente connesso su quella LAN ed in particolare con i router.

3. A questo punto se non ci sono router direttamente connessi sul link la procedura di stateless autoconfiguration termina, mentre se ne esiste almeno uno la procedura continua per l'assegnazione di un indirizzo "globale".

4. Se l'host riceve dei messaggi di router advertisement (RA), li interpreta e si autoconfigura (in questi messaggi sono contenuti i parametri fondamentali). Poiché l'intervallo di tempo che intercorre tra l'invio di due messaggi consecutivi da parte di un router è generalmente più lungo di quanto un host è disposto ad attendere, il nodo può inviare uno o più messaggi di router solicitation (RS) forzando la risposta anticipata di un router.

Dal momento che i router inviano periodicamente nuovi messaggi di router advertisement, i nodi ricevono continuamente nuove informazioni con le quali aggiungere nuovi indirizzi o aggiornare quelli già configurati.

La procedura di autoconfigurazione stateless presenta due problemi che non sono stati ancora risolti: essendo gli indirizzi unicast costruiti a partire dall'identificativo di interfaccia, il cambio della scheda di rete comporta il cambio dell'indirizzo IP; inoltre la presenza su una rete di due router porta all'assegnazione di due indirizzi diversi alla stessa interfaccia.

4.5 MIPv6

Il funzionamento del MIPv6 si basa sull'idea di mantenere attiva la connessione senza interrompere la sessione attiva anche quando questo si muove nella rete.

Perchè questo sia possibile il protocollo sfrutta l'utilizzo di particolari indirizzi, componenti e messaggi.

4.5.1 Indirizzi

- l'**Home Address** (**HoA**): l'indirizzo che identifica il MN indipendentemente dalla sua posizione fisica e utilizzato dagli host (nel nostro caso

il correspondent node) che vogliono comunicare con esso. Corrisponde all'indirizzo con il quale il MN si connette alla rete wireless di partenza (denominata **Home Network**);

- il **Care-of Address** (**CoA**): l'indirizzo che il MN utilizza, dopo essersi spostato dalla home network, per comunicare con il suo Home Agent (il router della rete iniziale). Cambia in base al prefisso della rete wireless che in quel momento ospita il nodo mobile (detta anche **Foreign Network**).

4.5.2 Componenti

Per implementare il MIPv6 è necessario aggiungere alcune funzionalità al nodo mobile, al correspondent node e al router della Home Network. Questi prendono rispettivamente il nome di:

- **Mobile Node** (**MN**): è un semplice host a cui viene aggiunto il supporto per gestire il protocollo MIPv6 e la **Binding Update List**. Quest'ultima è una struttura dati che permette al MN di tener traccia dei nodi da aggiornare periodicamente riguardo il suo attuale CoA.

- **Home Agent** (**HA**): rispetto ad un normale router, supporta il protocollo MIPv6 e gestisce la **Binding Cache**, una struttura dati per tener traccia del CoA attuale del MN. Riveste l'importante ruolo di intermediario del traffico dati da/per il MN, rendendo così trasparenti gli spostamenti di quest'ultimo agli altri nodi.

- **Correspondent Node** (**CN**): è un normale host fisso al quale, nel caso in cui è previsto il Return Routability, viene aggiunto il supporto per gestire il MIPv6 e la Binding Cache.

4.5.3 Messaggi

Per la gestione del binding:

- Binding Update (BU)

- Binding Acknowledgement (BAck)
- Binding Refresh Request (BRR)
- Binding Error (BE)

Per gestire l'autenticità del binding:

- Home Test Init (HoTI)
- Care-of Test Init (CoTI)
- Home Test (HoT)
- Care-of Test (CoT)

4.5.4 Funzionamento

Il Mobile Node è connesso alla Home Network Nella fase iniziale, il dispositivo mobile si connette ad una rete wireless (che da quel momento in poi viene riconosciuta come la sua Home Network) e, utilizzando il suo attuale indirizzo (HoA), instaura una comunicazione VoIP con il correspondent node posizionato in una rete differente.

Finchè il MN rimane connesso alla sua Home Network, i pacchetti inviati all'HoA vengono instradati normalmente al MN.

Il Mobile Node entra in una Foreign Network Nel momento in cui si sposta in un'altra rete wireless, diventa raggiungibile tramite un nuovo indirizzo, il CoA, con lo stesso prefisso della Foreign Network che lo ospita.

A questo punto il MN, per continuare a ricevere i pacchetti del CN, notifica all'Home Agent il suo nuovo indirizzo (in termini tecnici, il "MN si registra presso l'HA" o "crea una **binding** con l'HA"), inviando un messaggio BU. L'HA aggiorna la Binding Cache e notifica al nodo mobile l'avvenuta ricezione del BU ripondendo con un BAck. Da questo momento in poi l'Home Agent

sarà in grado, utilizzando la tecnica del tunneling IPv6[4], di svolgere il ruolo di intermediario del traffico dati da/per il MN.

In particolare, quando il CN invia pacchetti verso l'HoA del nodo mobile, questi giungono fino all'Home Agent, il quale, mediante un tunnel IPv6, provvederà ad inoltrarli presso l'attuale CoA del MN. Viceversa, i pacchetti del MN destinati al CN vengono prima inviati al HA tramite tunnel e poi normalmente instradati.

Route Optimization Per ottimizzare l'instradamento (**Route Optimization**) del traffico dati tra il MN e il CN, evitando il passaggio intermedio presso l'Home Agent (il cosiddetto **instradamento triangolare**), il MIPv6 permette di creare un canale diretto tra i due interlocutori. Per far ciò, è sufficiente che il mobile node notifichi il proprio CoA al CN per mezzo di un messaggio BU. Per garantire al CN una ragionevole sicurezza che il mittente del BU sia effettivamente il MN[5] (o perlomeno raggiungibile tramite entrambi gli indirizzi, l'HoA e il CoA), il Mobile IPv6 introduce un meccanismo di protezione chiamato **Return Routability**, basato sull'autenticazione dei messaggi.

Per avviare la suddetta procedura, il MN invia contemporaneamente due messaggi al CN: uno direttamente (il Care of Test Init, CoTI) e uno tramite l'Home Agent (il Home Test Init, HoTI).

Sulla base dei messaggi ricevuti, il CN genera una chiave e la inserisce nei due messaggi con i quali risponde al MN: uno lo invia direttamente (il Care of Test, CoT) e l'altro tramite l'HA (il Home Test, HoT).

A questo punto, il MN invia un BU al CN autendicandolo tramite una firma digitale formulata sulla base della chiave ricevuta; il CN, controlla l'autenticità

[4]Tunneling IPv6: in generale, è una tecnica per costruire linee di collegamento virtuali point-to-point fra una coppia di nodi che sono, in realtà, separati da un numero arbitrario di reti.
In questo caso viene utilizzato per costringere un pacchetto ad essere consegnato in un luogo particolare, anche se la sua destinazione originale potrebbe portare il pacchetto in un luogo diverso. Per creare l'illusione che il pacchetto viaggi in un tunnel, viene incapsulato in un nuovo pacchetto IPv6 prima di essere instradato, e decapsulato una volta giunto a destinazione. In questo modo, solo i nodi collegati dal tunnel sono in grado di leggere il pacchetto originale.

[5]Falsificando i BU, infatti, è possibile alterare i flussi di traffico, provocando attacchi DoS o man-in-the-middle.

del BU ricevuto, aggiorna la propria Binding Cache e notifica al nodo mobile l'avvenuta ricezione del BU ripondendo con un BAck.

Da questo momento in poi, i due nodi sono in grado di comunicare direttamente.

Per rendere trasparente la mobilità del nodo e per risolvere il problema legato all'ingress filtering alla Foreign Network[6], per lo scambio dei pacchetti tra MN e CN viene utilizzato un meccanismo più efficiente del Tunneling IP. Pertanto le trasmissioni avvengono nel seguente modo:

- I pacchetti spediti dal MN hanno come indirizzo sorgente il CoA ma contengono l'HoA all'interno di un Destination Option Extension Header[7] (l'Home Address Destination Option). Una volta a destinazione, vengono riformati con l'HoA, rendendo la procedura invisibile ai livelli superiori.

- i pacchetti inviati dal CN hanno come indirizzo sorgente l'indirizzo del CN, come destinazione il CoA e uno speciale Routing Header[8] (di tipo 2) contenente l'HoA. Arrivati nel MN, vengono riformati con con l'HoA, rendendo la procedura invisibile ai livelli superiori.

4.6 Interfacce Wireless Multiple

Su questa base è stato necessario introdurre una gestione di interfacce wireless multiple per MIPv6.

La gestione delle interfacce wireless da parte di INET è piuttosto limitata, poichè avviene attraverso una tabella statica dei dispositivi presenti nella rete (creata quando i nodi vengono inizializzati), invece che delle radio wireless. È

[6] I meccanismi di Network Ingress Filtering (RFC 2267) offrono una certa protezione contro attacchi di tipo Distributed Denial Of Service. Questo meccanismo infatti prevede che solo i pacchetti che abbiano un indirizzo sorgente appartenente allo spazio di indirizzamento relativo alla rete stessa possano essere inoltrati dal router verso il resto del mondo. Nel nostro caso, quindi, le trasmissioni dal MN al CN devono avere come indirizzo sorgente il CoA del MN

[7] Destination Option Header: uno dei possibili Extension Header dei pacchetti IPv6[**?**]; viene utilizzato per trasportare informazioni opzionali che devono essere esaminate solo dai nodi destinazione del pacchetto

[8] Routing Header: uno dei possibili Extension Header dei pacchetti IPv6[**?**]; contiene informazioni utili all'instradamento del pacchetto.

stato quindi necessario modificare questo meccanismo per permettere ad un host di contenere più intefacce wireless. Per ottenere questo risultato ci siamo basati sul lavoro svolto da Piero Murphy[21]

Per prima cosa è stato aggiunto il file HostEntry.h che ridefinisce una struttura dati (HostEntry), definita nei file per la gestione di dispositivi wireless di INET, per permettere la memorizzazione di tutte le informazioni sui dispositivi wireless necessarie.

Grazie a questa ridefinizione i file incaricati di gestire le connessioni wireless riusciranno a lavorare sulle singole radio e non sui singoli host. Per nodo wireless esisterà una struttura HostEntry contenente le infomazioni:

- modulo dell'host
- posizioni spaziale
- un numero di strutture RadioEntry pari al numero delle interfacce wireless

4.7 Multiple Care of Address

Nella sua implementazione attuale il protocollo MIPv6 permette la registrazione di un solo Care of Address, limitando a livello teorico la dinamicità dell'host, il multihoming e le performance di molte applicazioni in grado di creare più connessioni contemporaneamente.

Ogni nodo può avere molteplici accessi sia in entrata che in uscita contemporaneamente grazie alle numerose tecnologie disponibili, ed è quindi necessario che i protocolli per la gestione della mobilità siano in grado di supportarli. MIPv6 riesce già a mantenere una connessione costante anche in presenza di handover con un buon livello di qualità, ma risulta limitato dalla possibilità di avere un singolo Home Address ed un singolo Care of Address.

Per superare le limitazioni di MIPv6 creando nuova elasticità verso i fallimenti, visto che se una connessione non va a buon fine, se ne può usare un'altra che è già stata registrata precedentemente senza perdite, è stata creata una

sua estensione, chiamata MONAMI, che ne implementa il MCoA. Questo sistema produce anche alcuni svantaggi poichè non specifica nessun meccanismo per l'utilizzo dei molteplici CoA registrati, per esempio un'applicazione VoIP potrebbe essere interessata ad una connessione che riduca al minimo la latenza mentre un applicazione per il download necessiterebbe di un'alta capacità di trasmissione. Sarebbe inoltre necessario un sistema di condivisione delle informazioni tra i diversi protocolli per cominicare la disponibilità dei nuovi indirizzi.

Per registrare le diverse associazioni deve essere anche introdotto il Binding Identifier (BID) che le identifica. Sono inoltre necessarie delle modifiche alle strutture Binding Update List e Binding Cache, e ai messaggi Biinding Update e Binding Acknowledgement, per la corretta trasmissione dei MCoA e le rispettive registrazioni. I due metodi per la resgistrazione sono: registrazione di massa che prevede che indirizzi multipli siano registrati con un singolo BU; oppure nell'utilizzare un singolo BU per ogni singolo indirizzo.

Attualmente questo modo è l'unico che è supportato dal nodo corrispondente, per evitare la complessità di un'instradazione su un set di indirizzi. I nodi che partecipano al binding (HA e MN) devono entrambi supportare questa modalità o scambiarsi un messaggio BA per declinare l'offerta.

4.8 File

L'implementazione del sistema ha richiesto la modifica o creazione di diversi file, qui elencati:

- /scr/world/HostEntry.h
- /scr/linklayer/contract/RadioState.h
- /src/linklayer/mfcore/AirFrame.msg
- /src/linklayer/AbstracRadio.cc/.h
- /src/linklayer/GenericRadio.ned
- /src/world/ChannelAcces.cc/.h/.ned

- /src/world/ChannelControl.cc/.h/.ned
- src/networklayer/icmpv6 : IPv6NeighbourDiscovery.cc
- src/nodes/ipv6/ : WirelessStandardHost6.ned

Capitolo 5

Test

Passiamo ora ad esaminare le caratteristiche della simulazione.

Prerequisito necessario affinché le operazioni che verranno introdotte vadano a buon fine, è avere sul proprio sistema il framework OMNeT++ versione 4.1[10] correttamente compilato e configurato.

5.1 Simulazione

Tutti i file, relativi alla simulazione realizzata in questa tesi e al test effettuato su di essa, sono raccolti nella cartella `xmipv6/examples/ipv6/mobileipv6/voiceoverwifiRWMA/`.

All'interno troviamo i seguenti files:

- `VoWiFiNetworkCity6.ned` : definisce la struttura della rete realizzata
- `omnetpp.ini`: contiene i parametri utilizzati per eseguire il test sulla rete realizzata
- `run`: script per facilitare il lancio del test
- `path.xml`: utilizzato nel test per determinare il percorso che il mobile node compie all'interno della rete.

5.1.1 Compilare il codice

Per lanciare la prima volta la simulazione, occorre prima aggiornare il makefile. Per far ciò è sufficiente spostarsi nella cartella principale `xmipv6/`, eseguire il comando **make makemakefiles** (per creare il makefile) e di seguito il comando **make** (per compilare i sorgenti). Questa procedura dovrà essere ripetuta ogni qualvolta si creino nuovi file.

Nel caso in cui vengano modificati dei file, invece, sarà sufficiente ricompilare il codice, lanciando il comando **make** dalla cartella principale.

La creazione o la modifica di file .ned e .ini non comporta la ricompilazione del codice.

5.1.2 Testare la simulazione creata

Per eseguire un test sulla rete definita in `VoWiFiNetworkCity6.ned`, è necessario

1. spostarsi nella cartella `xmipv6/examples/ipv6/mobileipv6/voiceoverwifiRWMA/`
2. definire i parametri con i quali si vuole testare la rete nel file `omnetpp.ini`
3. lanciare lo script tramite il comando `./run`.

5.2 Test

Verranno ora illustrati i parametri utilizzati nel test e i risultati ottenuti.

5.2.1 Configurazione della rete

Per studiare le prestazioni dell'approccio simulato, dopo aver definito la struttura della rete basata sul protocollo MIPv6, è stato necessario effettuare un test con i seguenti parametri (verranno mostrati solo quelli di maggiore interesse):

- Dimensione dello scenario: $2000 * 1000\,m$

- Router:
  - Frequenza dei Router Advertisement: scelta con distribuzione uniforme nell'intervallo $[30, 70]ms$
- Canali radio:
  - AP: sintonizzati staticamente su un solo canale
    - AP_1: canale 1
    - AP_2: canale 3
    - AP_3: canale 6
    - AP_HOME: canale 9
    - AP_5: canale 11
  - Host wireless mobile: scansiona tutti i canali tra 1 e 11
  - Host wireless fissi: scansionano solo il canale dell'AP più vicino
- Host Mobile:
  - Applicazione: **DPBasicApp2**
  - Dimensione dei messaggi: $100B$
  - Frequenza dei messaggi: $40ms$
  - Mobilità:
    - Modulo: **TurtleMobility**
    - Percorso: descritto nel file `path.xml` (evidenziato in figura 6.1)
    - Velocità: scelta con distribuzione uniforme nell'intervallo $[1, 2]$ m/s per ogni segmento del percorso
- Correspondent Node:
  - Applicazione: **UDPBasicApp2**
  - Dimensione dei messaggi: $100B$
  - Frequenza dei messaggi: $40ms$

- Traffico background wired
  - Applicazione: **UDPBasicApp2**
  - Dimensione dei messaggi: $100B$
  - Frequenza dei messaggi:
    - (hostfix1$\Longleftrightarrow$hostfix6) : $122ms$
    - (hostfix2$\Longleftrightarrow$hostfix7) : $180ms$
    - (hostfix3$\Longleftrightarrow$hostfix8) : $156ms$
    - (hostfix4$\Longleftrightarrow$hostfix9) : $223ms$
    - (hostfix5$\Longleftrightarrow$hostfix10) : $100ms$
- Traffico background wireless
  - Applicazione: **UDPBasicApp2**
  - Dimensione dei messaggi: $100B$
  - Frequenza dei messaggi:
    - (whost1$\Longrightarrow$whost2) : $105ms$
    - (whost2$\Longrightarrow$whost3) : $92ms$
    - (whost3$\Longrightarrow$whost1) : $2ms$

5.2.2 Risultati

Per elaborare i risultati ho utilizzato il metodo delle prove ripetute, su simulazioni di 10 minuti, durante i quali il nodo mobile percorreva il suo tragitto per intero.

Per prima cosa analizziamo i risultati ottenutinella simulazione con il nodo mobile dotato di una sola interfaccia.

Come possiamo vedere dal grafico 5.2, in particolare esaminando la linea rossa che rappresenta il numero di pacchetti ricevuti dal nodo mobile quando questo e' dotato di una sola interfaccia di rete, le prestazioni non sono molto elevate. Il MN ha ricevuto 8214 pacchetti in media su 15014 pacchetti

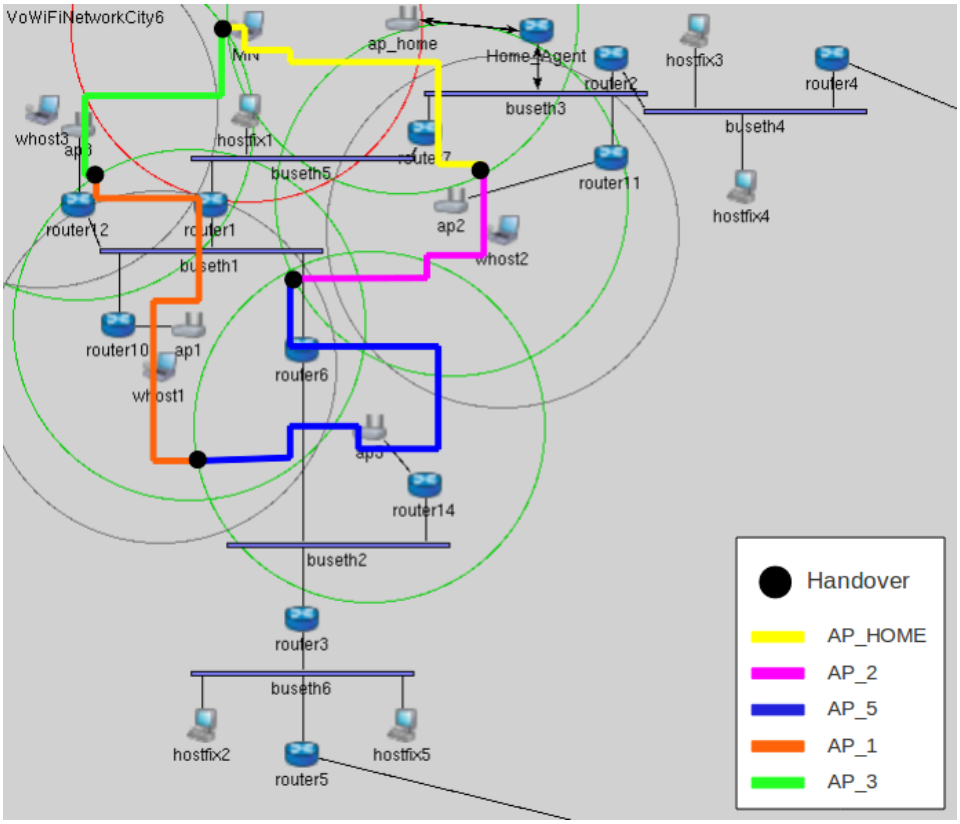

Figura 5.1: Percorso effettuato dal nodo mobile all'interno della rete

totali spediti dal CN (pari al 54,7%); mentre il CN ha ricavuto 7026 pacchetti su 15002 (pari al 46,8%). I principale motivo di questa prestazione non soddisfacente si può ricercare negli Handover.

Durante il passaggio da una rete all'altra infatti il nodo mobile diventa irraggiungibile e in quei secondi non riceve ne invia pacchetti. Un modo per ridurre l'icidenza di questo problema sarebbe quello di avere più intefacce wireless in modo che al cambio di rete la seconda interfaccia sia già configurata per inviare e ricevere dalla nuova rete.

Abbiamo quindi aggiunto una seconda scheda di rete per verificarne le prestazioni.

Dai risultati della simulazione ottenuti utilizzando 2 interfacce di rete, come rappresentato dalla linea gialla nel grafico 5.2, possiamo confermare i limiti di MIPv6dotato di più interfacce senza politiche per il Multiple Care of Address.

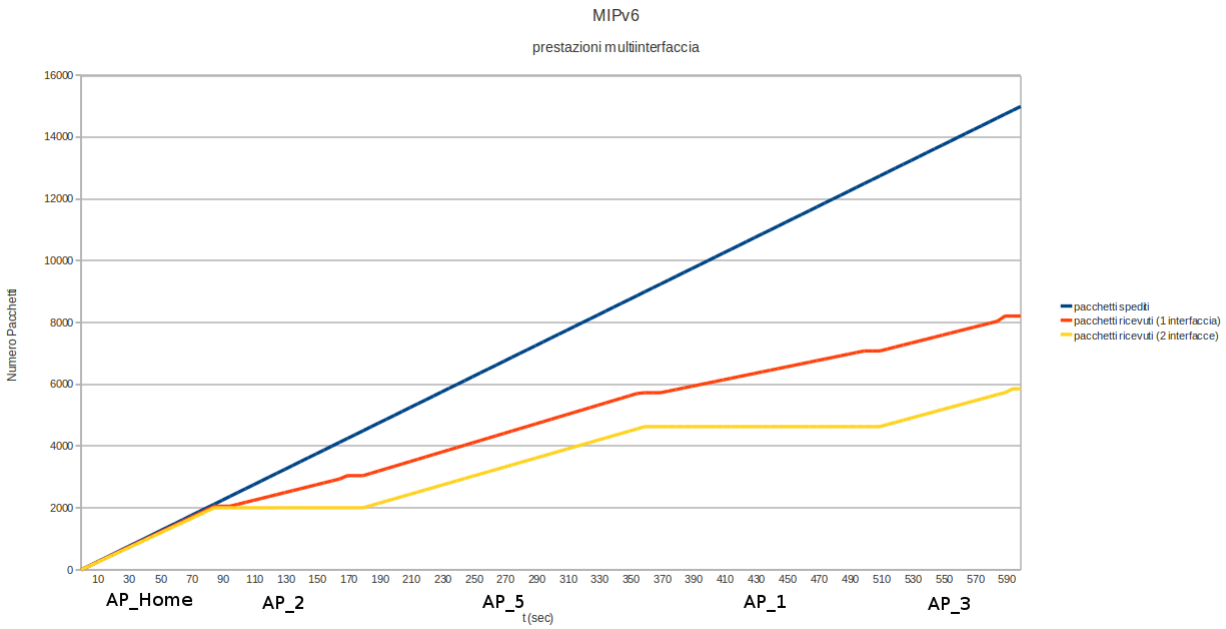

Figura 5.2: Grafico prestazioni MIPv6

All'avvio della simulazione la prima interfaccia si connette con l'AP_home e individua l'Home Agent correttamente senza presentare nessuna differenza ne calo di prestazioni rispetto all'utilizzo di una interfaccia singola. Quando si esce dal range dell'AP_home e le comunicazioni passano attraverso l'AP_2 cominciano a verificarsi i problemi dovuti alla mancanza di Multiple Care of Address.

La seconda interfaccia infatti si collega con l'AP_2 corretamente, ma non riesce a collegarsi con l'Home Agent in quanto non ha settato il Care of Address. Nel periodo di tempo il cui la seconda interfaccia resta attiva non vi è alcuna comunicazione tra mobile node e correspondent node.

Quando la connessione della seconda interfaccia cade e ritorna attiva la prima, la connessione con il correspondent node viene ristabilita correttamente finchè, restiamo all'interno del range del AP_5.

A ogni cambio di rete, cambia l'intefaccia connessa e si ripresenta il calo di prestazioni.

Il grafico mostra con chiarezza come i momenti in cui si verifica un cambio

di rete si interrompe o riprende la ricezione dei pacchetti.

Confrontando i risultati delle due simulazioni risulta chiaro che, in assenza di politiche per la gestione del Multiple Care of Address, l'introduzione della seconda interfaccia provochi addirittura un aumento delle perdite di pacchetti rispetto al caso in cui il nodo mobile sia dotato di una sola interfaccia.

Capitolo 6

Conclusioni

L'obiettivo di questa tesi era di produrre un contesto per la simulazione di MIPv6 realistico e credibile, e di confrontarlo con altre architetture esistenti per confrontarne le prestazioni.

Con questo scopo ci siamo impegnati utilizzando l'ambiente simulativo OMNeT++ ed il framework xMIPv6 a creare una simulazione che rispecchiasse un contesto reale; modificandolo per ampliarne le possibilità di sviluppo.

Sono state quindi eseguite delle simulazioni per verificare il comportamento delle architetture proposte per comprenderne i pregi e i limiti.

Il protocollo MIPv6, su cui ci siamo concentrati, si è dimostrato non soddisfacente a livello prestazioni per l'utilizzo di applicazioni VoIP. Le simulazioni mostrano infatti delle prestazioni non sufficenti per applicazioni per cui è rischiesta un bassa percentuale di pacchetti persi.

Abbiamo quindi cercato di migliorarne le prestazioni introducendo una seconda interfaccia di rete che aumentasse le prestazioni riducendo le perdite dovute ad Handover. La simulazione ha però evidenziato il problema legato alla mancanza di una politica standard per la gestione del Multiple Care of Address, mostrando i limiti del protocollo MIPv6. La simulazione ha dato infatti risultati peggiori rispetto all'utilizzo di una singola interfaccia.

In conclusione il nostro lavoro è un ottima piattaforma per eseguire sviluppi futuri tesi a migliorare ancora la credibilità e precisione della simulazione, introducendo una politica di gestione di più intefacce per Multiple Care of

Address.

Capitolo 7

Sviluppi futuri

7.1 Multiple Care of Address

Il framework xMIPv6 non ha introdotto nessun sistema per la gestione di più interfacce wireless su un nodo mobile MIPv6; sarebbe quindi opportuno implementare la gestione del Multiple Care of Address.

7.2 Ostacoli

Nella simulazione realizzata, la potenza del segnale radio viene valutata esclusivamente in base alla distanza. Sarebbe opportuno, per ottenere un maggiore realismo nelle comunicazioni wireless, introdurre nello scenario degli ostacoli (come ad esempio le pareti di un edificio) che interferiscano negativamente sulla potenza del segnale.

Un tale modello simulativo è già stato realizzato da Marco Pattaro e Francesca Montevecchi [11] e utilizzato da Marco Ciaramella [**?**] nello scenario VoWiFiNetworkCity.

7.3 Scelta dell'access point

La versione del framework xMIPv6 utilizzata nella tesi, riguardo la scelta dell'access point da parte di un dispositivo mobile, fornisce una soluzione non

sempre conveniente. La scelta infatti viene effettuata in base al primo access point trovato, mentre sarebbe più opportuno che la decisione venga presa tenendo in considerazione la distanza tra il nodo e l'AP e sopratutto la potenza del segnale ricevuto.

7.4 Home agent multipli

Testando la rete implementata, è stato possibile notare che, nel caso in cui nella rete ci siano più home agent, il nodo mobile non è in grado di distinguere la sua vera home network dalle foreign network dotate di un home agent; il risultato è che il nodo, anzichè configurarsi con un nuovo Care-of Address, riconfigura il proprio Home Address utilizzando il prefisso dell'home agent presente nella foreign network.

Nel caso si voglia gestire una rete con più nodi mobili, è necessario prima risolvere questo problema.

7.5 Point-to-Point Protocol (PPP)

Un altro problema riscontrato in xMIPv6 è che i frame inviati tramite un collegamento PPP non vengono correttamente configurati prima di essere inviati (in particolare sembra che non venga settato l'indirizzo MAC del destinatario del frame) e di conseguenza scartati dal nodo che li riceve.

Nel caso si voglia gestire una rete contenente collegamenti PPP, è necessario prima risolvere questo problema.

Bibliografia

[1] J. Postel, "RFC768 - User Datagram Protocol", Website, 1980, http://www.faqs.org/rfcs/rfc768.html

[2] Information Sciences Institute University of Southern California, "RFC791 - Internet Protocol", Websiste, 1981, http://www.faqs.org/rfcs/rfc791.html

[3] IEEE Std. 802.11b, "Higher-Speed Physical Layer (PHY) extension in the 2.4 GHz band", IEEE Standard for Information Technology, 1999

[4] IEEE Std. 802.11g, "Further Higher-Speed Physical Layer Extension in the 2.4 GHz Band", IEEE Standard for Information Technology, 2003

[5] IEEE Std. 802.11n, Website, 2009, http://standards.ieee.org/announcements/ieee802.11n_2009amendment_ratified.html

[6] IEEE Std. 802.11e, Website, 2008, http://standards.ieee.org/getieee802/download/802.11e-2005.pdf

[7] ANSI/IEEE Std 802.11, 1999 Edition

[8] IEEE Std. 802.11n, "Higher Throughput Improvements using MIMO", IEEE Standard for Information Technology, 2007

[9] OMNet++ staff, OMNet++ v4.1 User Manual, Website, http://www.omnetpp.org/doc/omnetpp41/manual/usman.html

[10] OMNet++ v4.1 e precedenti, http://www.omnetpp.org/omnetpp

[11] Marco Pattaro e Francesca Montevecchi, "Scenario simulativo per applicazioni VoIP su WiFi", Università degli Studi di Bologna, Non pubblicato

[12] V. Ghini, G. Lodi, F. Panzieri, "Robust Wireless Medium Access for VoWLAN Applications: A cross level QoS Mechanism", Università degli Studi di Bologna, Non pubblicato

[13] Vittorio Ghini, "MIPv4 & MIPv6 - Overview of IP Mobility Protocols", http://www.cs.unibo.it/~ghini/didattica/sistdistrib/MIPv4_MIPv6.pdf

[14] Silvia Ripa, "Architetture per la mobilità", Università degli Studi di Bologna, Non pubblicato

[15] xMIPv6 per OMNeT++ 4.x, download sorgenti, Website, https://github.com/zarrar/xMIPv6/archives/master

[16] xMIPv6 per OMNeT++ 4.x, sito ufficiale, Website, http://www.kn.e-technik.tu-dortmund.de/content/view/232/lang,de/

[17] S. Thomson, T. Narten, T. Jinmei, "IPv6 Stateless Address Autoconfiguration", Request for Comments (Draft Standard) 4862, September 2007

[18] R. Droms, et al, "Dynamic Host Configuration Protocol for IPv6 (DHCPv6)", Request for Comments (Draft Standard) 3315, July 2003

[19] D. Johnson, C. E. Perkins and J. Arkko, "Mobility Support in IPv6", Request for Comments (Proposed Standard) 3775, Internet Engineering Task Force, June 2004

[20] "Internet Protocol versione 6: aspetti avanzati", Website, http://netgroup.polito.it/netlibrary/ipv6-adv/text.htm

[21] Piero Murphy, “Uno Strumento Simulativo per Architetture VoIP per Dispositivi Mobili Multihomed”, Università degli Studi di Bologna, Non pubblicato

[22] R. Kuntz, "Deploying Reliable IPv6 Temporary Networks thanks to NEMO Basic Support and Multiple Care-of Addresses Registration", 2007 International Symposium