Alma Mater Studiorum · Università di Bologna

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

Scenario simulativo per VoIP da dispositivi mobili basato su proxy server SIP/RTP

Tesi di Laurea in Reti di Calcolatori

Relatore:
Dott.
Vittorio Ghini

Presentata da:
Marco Ciaramella

Sessione I
Anno Accademico 2009/2010

Introduzione

Negli ultimi anni, grazie alla sempre più rapida evoluzione e diffusione di dispositivi mobili, come *smartphone* e *netbook*, insieme alla ormai sempre maggiore diffusione di punti d'accesso (AP) a reti **wireless** pubbliche nei centri urbani, si è resa realistica l'ipotesi di un passaggio verso sistemi di connettività per dispositivi mobili che sfruttino tecnologie **WiFi** (IEEE 802.11a/b/g/n). In particolare, risulta interessante la possibilità di utilizzare applicazioni **Voice over Internet Protocol** (VoIP) su questo tipo di reti, in contesti di mobilità su scala urbana. Allo scopo di mantenere un livello di **Quality of Service** (QoS) accettabile per applicazioni di telefonia in questo tipo di scenari, è stato realizzato il meccanismo **Robust Wireless Medium Access**[**?**] (RWMA) nel kernel del sistema operativo open-source GNU/Linux.

Come per molte nuove tecnologie, anche per il meccanismo RWMA, sono richiesti dei test, spesso troppo complessi o costosi per essere eseguiti con dispositivi reali, per verificarne le condizioni di applicabilità, le prestazioni e l'affidabilità; per questo motivo si procede a sviluppare una simulazione del meccanismo, una versione virtuale, che tenga conto dei fattori rilevanti per quanto riguarda gli indici di prestazione che si è scelto di esaminare. In particolare questa tesi si occupa della realizzazione di un modello di propagazione del segnale radio tra diversi dispositivi Wifi, in presenza di ostacoli. A tale scopo si perfezionerà un modello preesistente di misurazione della potenza del segnale ricevuto da una stazione WiFi in modo da aumentarne l'affidabilità, inoltre sarà realizzata una semplice rappresentazione di ostacoli alla trasmis-

sione radio, grazie ai quali verrà infine definito uno scenario simulativo che rappresenti una stazione WiFi mobile, che si sposta tra le vie di un centro cittadino in presenza di vari AP eseguendo un'applicazione VoIP grazie al meccanismo RWMA.

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

[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

Access Point) che sono utilizzati per comunicare all'interno di una stessa BSS.

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 Infrastructure 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

esistono di due tipi: Independent Basic Service Set e Infrastructure Basic Service Set.

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

[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.

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.

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

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).

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

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 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[**?**].

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[**?**].

IEEE 802.11 n Lo standard è stato finalizzato l'11 Settembre 2009[**?**] 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

[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.

[8] Quantità di dati trasmessa in un determinato intervallo di tempo.

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):

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] Media Access Control (MAC). E' un sotto-livello del livello Data Link.

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

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

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.

[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.

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, dai pacchetti ricostruisce 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.

[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).

- 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.

1.3.3 Session Initiation Protocol

Il protocollo SIP è un protocollo del livello applicazione che assomiglia, in qualche modo, al protocollo HTTP, essendo basato su un modello richiesta/risposta simile, pur essendo stato progettato per applicazioni di tipo piuttosto diverso per cui ha potenzialità abbastanza diverse da quelle di HTTP. Le funzionalità messe a disposizione da SIP si possono raggruppare in cinque categorie:

- Raggiungibilità dell'utente: determinare il dispositivo corretto con cui comunicare per raggiungere un certo utente;
- Disponibilità dell'utente: determinare se l'utente vuole prendere parte ad una particolare sessione di comunicazione oppure se è in grado di farlo;
- Potenzialità dell'utente: determinare i media utilizzabili e i relativi schemi di codifica;
- Instaurazione della sessione: stabilire i parametri della sessione, come i numeri di porta, che devono essere utilizzati da entrambe le parti coinvolte nella comunicazione;

- Gestione della sessione: un ampio Instaurazione della sessione: stabilire i parametri della sessione, come i numeri di porta, che devono essere utilizzati da entrambe le parti coinvolte nella comunicazione.spettro di funzioni, che comprendono il trasferimento di sessioni (per realizzare, ad esempio, il "trasferimento di chiamata", call forwarding) e la modifica dei parametri di sessione;

1.3.4 User Datagram Protocol

Lo User Datagram Protocol[13][**?**] (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 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.

[13] Il protocollo e' 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.

- 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.

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.

1.3.5 Internet Protocol

Internet Protocol[16][**?**] (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.

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

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

Esistono due versioni del protocollo IP:

- Internet Protocol Version 4: Gli indirizzi sono a 32 bit, e per questo vi sono solo 4.294.967.296 indirizzi possibili. Un indirizzo consiste in una quadrupla di numeri separati da un punto (es: 192.168.0.1).
- Internet Protocol Version 6

Header IP La struttura di un header IPv4 è la seguente:

1. Versione (4 bit) : specifica se Ipv4 o IPv6
2. Internet Header Length (4 bits)
3. Tipo di servizio (8 bits): anche conosciuto come QoS, descrive che priorità deve avere il pacchetto
4. Lunghezza (16 bit): viene espressa in bytes
5. Identificazione (16 bit): aiuta a ricostruire il pacchetto dai frammenti
6. Flag Reserved (1 bit): sempre valorizzato con 0
7. Don't fragment (1bit) [DF]: valorizzato con 1 se il pacchetto può essere frammentato

8. More Fragments (1 bit) [MF]: valorizzato con 1 se seguono altri frammenti del datagram
9. Fragment offset (13 bit)
10. Time to Live (8 bit) [TTL]: numero di hop[17] attraverso il quale il pacchetto può passare prima di essere scartato
11. Protocollo (8 bit): TCP, UDP, ICMP, ecc...
12. Checksum (16 bit)
13. Indirizzo IP Mittente (32 bit)
14. Indirizzo IP Destinatario (32 bit)

Un datagram IP se necessario può essere frammentato a seconda dell'MTU[18] del mezzo che si vuole utilizzare.

Per ricostruire un datagram IP frammentato si utilizzano principalmente due campi: il bit frammentazione e l'offset. Di ogni datagram IP (con lo stesso identificatore), bisogna raccogliere tutti i frammenti fino ad avere quello con il campo moreFrag uguale a 0. A quel punto i frammenti possono essere assemblati nell'ordine corretto utilizzando gli offset come discriminante.

[17]Dispositivi di qualsiasi tipo (router, computer, ecc) attraverso i quali verrà instradato il pacchetto.

[18]Maximum transmission unit (lett: Unità massima di trasmissione) è la dimensione massima che può avare un pacchetto per viaggiare su una data rete.

Capitolo 2

Obiettivi

Il lavoro svolto nella tesi consiste nel ricreare, usando il simulatore OMNeT++, una comunicazione VoIP tra un end-sytem wireless mobile e un'altro fisso o mobile situati in due reti diverse. I due end-system operano in uno scenario cittadino ed utilizzano un'architettura denominata RWMA(Robust Wireless Medium Access) che fornisce supporto per la QoS[1].

L'architettura 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.

[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.

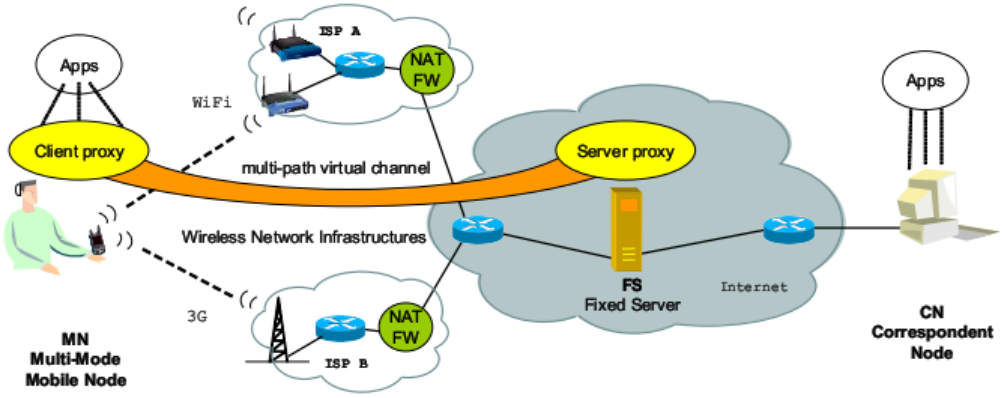

Figura 2.1: Possibile scenario di utilizzo

2.1 Modello QoS: Robust Wireless Medium Access

Il meccanismo di Quality of Service chiamato Robust Wireless Medium Access garantisce le seguenti due proprietà nelle comunicazioni VoIP:

1. Perdita dei pacchetti inferiore al 3% di quelli trasmessi

2. Un 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.

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

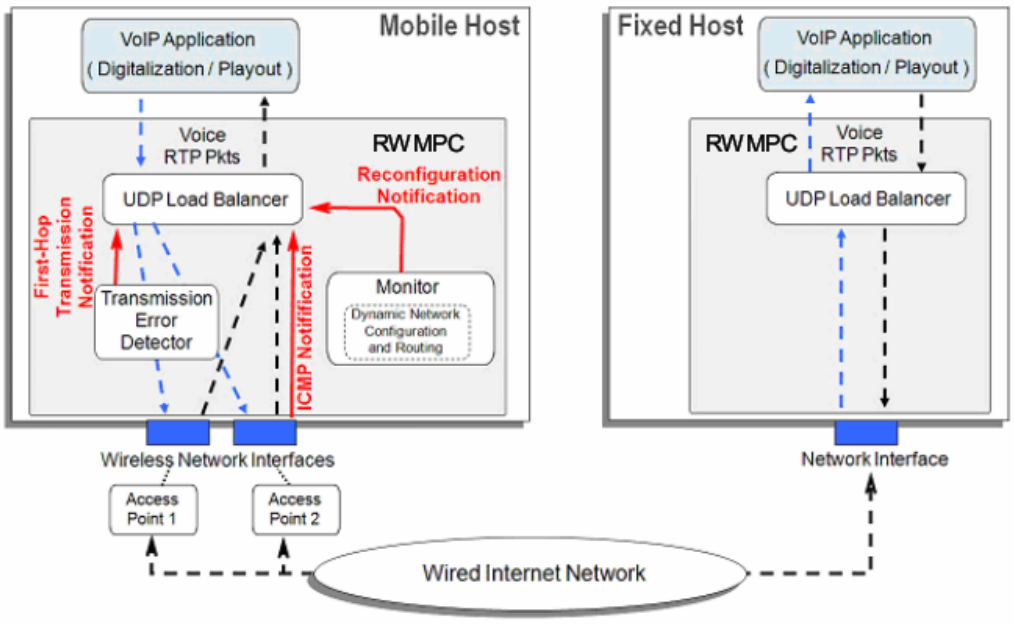

Figura 2.2: Architettura del sistema RWMA

2.1.1 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[**?**, **?**, **?**, **?**].

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.

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.

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 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

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

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.

2.1.3 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 del sistema RWMA su Linux

L'architettura RWMA è stata implementata con successo su un sistema Linux (Gentoo) con un kernel versione 2.6.27.4.

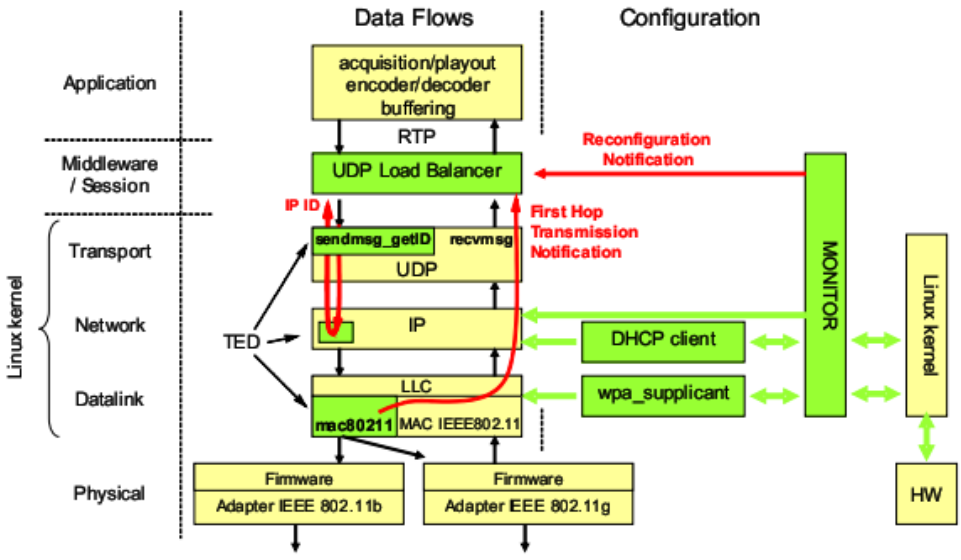

Figura 2.3: Implementazione RWMA su Linux

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

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 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.

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.

[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.

[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.

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.

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

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

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

2.3 Problemi

2.3.1 Firewall e sistemi NAT nelle reti wireless

L'ampia diffusione di firewall e sistemi NAT, ai margini della maggior parte delle reti wireless accessibili, impediscono ai nodi esterni alla rete di

avviare una comunicazione con nodi interni. In una situazione del genere l'unico modo per instaurare una comunicazione è usare un server esterno, al di fuori da qualsiasi firewall e sistema NAT, che agisca come **instradatore** (relay) per i dati.

2.3.2 Identità degli host mobili all'interno di Internet

I nodi wireless, essendo liberi di muoversi, possono in qualsiasi momento cambiare rete, perdendo di conseguenza identità nell'intera Internet. Per fare in modo che il nodo continui ad essere raggiungibile, è necessaria la presenza di un server DHCP noto al nodo mobile e che permetta una configurazione automatica dell'host ogni volta che questo si collega ad una nuova rete.

2.4 Obiettivi della tesi

Gli obiettivi che ci si è prefissati di raggiungere sono:

1. Costruire una rete che riproponga l'infrastruttura di Internet. La rete dovrà essere un'interconnessione di più sottoreti in cui sono presenti host di reti diverse che comunicano tra loro, router che instradano pacchetti fra le diverse reti, e access point che consentono agli host wireless di connettersi alla rete. Inoltre si cercherà di riprodurre situazioni reali presenti in una rete, come traffico generico di background, **backbone**[7] di collegamento ad alta latenza, e corruzione dei dati nelle trasmissioni wireless. L'implementazione farà uso dei moduli sviluppati da Marco Pattaro e Francesca Montevecchi [**?**] per introdurre nello scenario ostacoli ed avere un modello di perdita del segnale radio più realistico, ottenendo così un maggiore realismo nelle comunicazioni wireless.

2. Implementare un **proxy server** che utilizzi l'infrastruttura SIP (Session Initiation Protocol) per raggirare i **firewall** e i **sistemi NAT** delle

[7] Letteralmente spina dorsale, in Internet è un insieme di percorsi tra nodi che permettono interconnessioni a lunga distanza a reti locali e regionali.

reti wireless e per garantire la raggiungibilità dei nodi mobili quando questi cambiano rete. Il server dovrà essere situato al di fuori della rete wireless in questione.

3. Implementare un protocollo DHCP che permetta al nodo mobile di cambiare indirizzo IP ogni volta che questo si sposta in una sottorete diversa.

2.5 Scelte implementative

Al fine di raggiungere efficacemente gli obiettivi preposti sono state prese le seguenti decisioni:

- la parte fissa della comunicazione VoIP si è pensata di implementarla direttamente sul server proxy che fa quindi le veci del secondo end system.
- il protocollo DHCP è stato implementato nel livello applicazione dell'infrastruttura RWMA. In particolare si è deciso di inserirlo all'interno del load balancer per motivi prestazionali.
- a differenza della versione originale del protocollo DHCP, in quella implementata non avviene alcuna trasmissione broadcast delle richieste DHCP ma invece è noto, all'host che si vuole configurare, l'indirizzo IP del server DHCP. In pratica l'host funziona anche da agente di collegamento[8].

2.6 Ambiente simulativo

Si è scelto di sviluppare l'applicazione utilizzando la piattaforma simulativa OMNeT++, che fornisce il sistema di gestione degli eventi, sulla quale

[8]Nella versione originale del protocollo, l'agente di collegamento è un nodo della rete incaricato di comunicare con il server DHCP attraverso una comunicazine unicast dopo aver ricevuto una richiesta broadcast dall'host.

è stato esteso il framework INET, che implementa i più diffusi protocolli di rete.

Capitolo 3

Framework

In questo capitolo verranno introdotti i concetti fondamentali necessari per 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++[**?**] e INET[**?**], 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 del meccanismo RWMA.

La versione di OMNeT++ a cui facciamo riferimento è la 4.0 (stabile) rilasciata il 27 Febbraio 2009[**?**].

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.

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

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.

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 vendono 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 mo-

duli. Gli oggetti control info possono ovviamente essere definiti dall'utente, e sono sottoclassi di cObject (libreria OMNeT++).

3.1.5 Il 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 I 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.

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

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

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 INET

Il framework INET 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.

Contiene modelli per diversi protocolli tra i quali citiamo: UDP, TCP, SCTP, IP, IPv6, Ethernet, PPP, IEEE 802.11, MPLS, OSPF.

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 INET 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: RoutingTable), gestiscono la comunicazione (es: Notification Board), l'auto-configurazione di un network (es: FlatNetworkConfigurator), il movimento dei nodi (es: LinearMobility), 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 inter-

facce (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.

- RoutingTable: questo modulo gestisce le routing table per IPv4 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.

3.2.3 Architettura di una NIC IEEE 802.11

Una interfaccia NIC (Ieee80211) nel framework INET 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

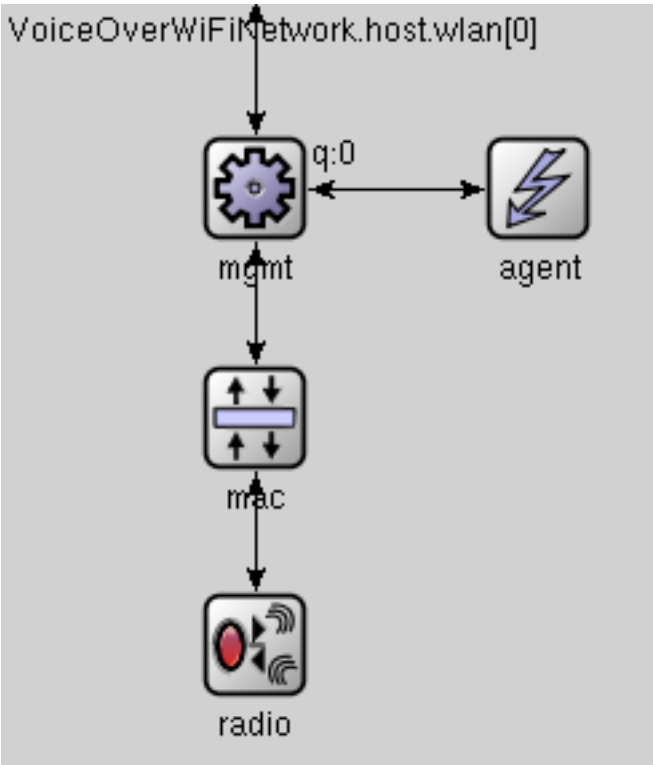

Figura 3.1: Architettura di una WNIC in INET

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 I file .irt

I file di Routing hanno estensione irt, e vengono utilizzati per configurare il modulo RoutingTable (presente in tutti i nodi IP) prima del lancio di una simulazione.

Questi file possono contenere la configurazione delle interfacce di rete e delle rotte statiche (che verranno aggiunte alla routing table), entrambe opzionali.

Le interfacce sono identificate da nomi (es: ppp0, ppp1, eth0) originariamente definiti nel file NED.

Ogni file è suddiviso in due sezioni (che per struttura ricordano l'output dei comandi "ifconfig" e "netstat -rn" di Unix):

- Compresa tra le keyword "ifconfig" e "ifconfigend": Configurazione delle interfacce
- Compresa tra le keyword "route" e "routeend": Contiene le rotte statiche

Interfacce I campi accettati nel definire un'interfaccia sono:

- name: nome (e.g. ppp0, ppp1, eth0)

[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.

- inet_addr: indirizzo IP
- Mask: netmask
- Groups: gruppo Multicast
- MTU: MTU del canale
- Metric
- flag: BROADCAST, MULTICAST, POINTTOPOINT

Rotte I campi accettati nelle rotte sono:

- Destination: Indirizzo IP del destinatario (o default)
- Gateway
- Netmask
- Flags: H (host: rotta diretta per il router) o G (gateway, rotta remota attraverso un altro router)
- Metric
- Interface

3.2.5 Le simulazioni

Le simulazioni in OMNeT++/INET 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 INET con il file come parametro.

- /PATH_TO_INET/run_inet <*fileDiConfigurazione.ini*>

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_INET/run_inet -c <*nomeDellaSimulazione fileDiConfigurazione.ini*>

3.3 Architettura RWMA

3.3.1 ULB

Il cuore del meccanismo RWMA, ovvero il modulo dove tutti i messaggi di notifica vengono inviati per essere analizzati, è stato implementato a Livello Applicazione. Il modulo nel quale è stata implementata la componente ULB è chiamato ULBRWMA (.cc/h/ned) ed è una sottoclasse di UDPBasicApp[4]. Una qualsiasi applicazione che desideri usufruire dei nostri servizi, deve necessariamente essere una sottoclasse di questo modulo.

3.3.2 Richiesta del servizio

Quando un'applicazione vuole inviare un pacchetto (RTP o di qualsiasi altro tipo) usufruendo del servizio RWMA, deve utilizzare la chiamata sendToUDPRWMA()[5] ereditata dalla classe ULBRWMA.

[4]UDPBasicApp è una sottoclasse di UDPAppBase, e si occupa di gestire i comportamenti base di tutte le applicazioni UDP. Non possiede un gran numero di funzioni, ma contiene in se già i campi basilari necessari a gestire una comunicazione UDP, come le porte o gli indirizzi.

[5]void sendToUDPRWMA (cPacket *msg, const IPvXAddress &srcAddr, int srcPort, const IPvXAddress &destAddr, int destPort).

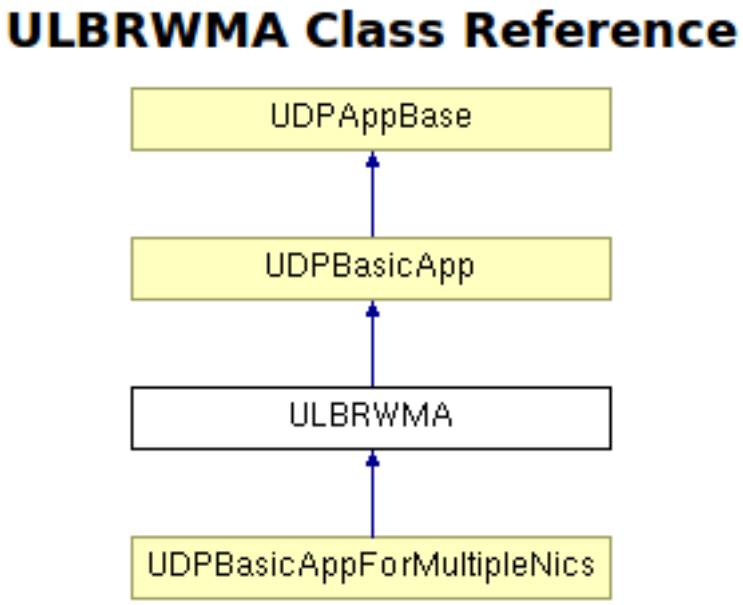

Figura 3.2: ULBRWMA Class Reference

Come comprensibile dal nome della funzione, questa si comporta in modo analogo alla chiamata sendToUDP()[6], che è la funzione originale utilizzata per inviare un pacchetto con il protocollo UDP. La versione implementata nella tesi si differenzia da quest'ultima perché, oltre ad occuparsi di inviare il pacchetto, lo configura affinché sia riconoscibile dai livelli inferiori come pacchetto RWMA.

Prima di inoltrarlo, una sua copia viene inserita in una coda, in attesa che il livello IP ci faccia pervenire il suo identificatore.

La chiamata sendToUDPRWMA(), al contrario della sua controparte realizzata su Linux, **non è bloccante**. Questo è dovuto al fatto che in OMNeT++, che ricordiamo è basato su un meccanismo di message passing, l'unico modo che abbiamo per richiedere un servizio/informazione attraverso i vari livelli (simili a comparti stagni), è fare una richiesta con un pacchetto

[6]void sendToUDP (cPacket *msg, int srcPort, const IpvXAddress &destAddr, int destPort) è una funzione definita in UDPAppBase.

adeguato ed aspettare la risposta.

La funzione inoltre, permette di specificare tutti i parametri necessari alla trasmissione: pacchetto, porte e indirizzi (mittente e destinazione). Questi valori possono essere ottenuti grazie a delle funzioni di aiuto, che si occupano di recuperarli e restituirli (questi parametri devono necessariamente essere stati specificati nel file di configurazione ini).

UDP senza RWMA L'applicazione non è ovviamente obbligata ad usare il nostro servizio. Se desidera utilizzare il protocollo standard, può inviare i suoi pacchetti attraverso la chiamata sendToUDP(). Questo evita che l'applicazione sia costretta sempre e comunque a richiedere un servizio che magari non è realizzabile (es: il nodo non sia connesso attraverso una rete wireless, ma su cavo).

3.3.3 Identificatore Datagram IP

Il pacchetto viaggia attraverso il modulo UDPRWMA (rimanendo sostanzialmente invariato), fino ad arrivare a quello IP. Qui la classe IPRWMA (.cc/h/ned) sottoclasse di IP (.cc/h/ned), accorgendosi della richiesta di inoltrare il pacchetto ai livelli inferiori, lo controlla per verificare se è stato richiesto il servizio.

Per verificarlo, viene controllato il valore del campo protocol (protocollo) del pacchetto. Se questo è uguale a IP_PROT_UDP[7] (17), il pacchetto è stato inviato attraverso il protocollo UDP in maniera standard e viene inoltrato senza ulteriori operazioni, seguendo il normale flusso previsto per tutti i pacchetti.

Se questo valore è invece IP_PROT_UDP_RWMA (300), il pacchetto ha richiesto il servizio RWMA. A questo punto, il modulo invia un pacchetto di tipo IPNotify all'ULB, nel quale si notifica che questo è arrivato con successo al modulo IP, e ne restituisce l'identificativo univoco.

Il pacchetto scende successivamente di livello fino ad arrivare al WNIC.

[7]Per l'elenco dei valori standard ammessi, consultare il file IPProtocolId.msg.

3.3.4 MAC e RWMA

Quando un datagram IP arriva al MAC del WNIC, questo si appresta ad essere inviato attraverso l'interfaccia fisica su canale wireless.

Il protocollo IEEE 802.11 è correttamente implementato su INET, compreso il sistema di notifica della corretta ricezione di un frammento da parte dell'AP **attraverso il frame di controllo ACK**. Inoltre vengono anche gestiti autonomamente dei timer (con l'uso di self-message), in modo che il livello MAC consideri il pacchetto perso dopo un certo periodo di tempo senza aver ricevuto risposta.

Per sapere se il pacchetto ha richiesto il nostro servizio, nell'implementazione su linux il sistema si occupa di controllare se il socket UDP da cui proviene è correttamente valorizzato per ricevere le notifiche. Nel nostro caso invece, questo approccio non è implementabile. Essendo tutto OMNeT++ costruito a comparti stagni, il modulo MAC non ha modo di accedere ai socket presenti in quello UDP.

Il sistema utilizzato per risolvere il problema, è identico a quello utilizzato nel modulo IP, ovvero controllare il campo protocol del pacchetto.

3.3.5 TED

Riconosciuti i nostri pacchetti, il modulo che implementa il TED, chiamato Ieee80211MacRWMA (.cc/h/ned), sottoclasse di Ieee80211Mac (.cc/-h/ned), si occupa di monitorare due particolari eventi:

1. Viene ricevuto l'ACK inviato dall'Access Point

2. Sono passati il numero massimo di retry stabilito senza aver ricevuto l'ACK

Purtroppo in INET, il sistema non si occupa di segnalare l'avvenuta trasmissione di un frammento ai livelli superiori come invece dovrebbe (il meccanismo è segnalato come "da implementare", ed è possibile che venga affrontato dai sviluppatori di INET in un prossimo futuro).

Il TED si occupa quindi di segnalare entrambi gli eventi, creando un pacchetto IPNotify contenente le informazioni necessarie (che vengono ricavate dal pacchetto inviato), per poi spedirlo ai livelli superiori.

```
** Event #282168  T=9.119234054007  VoiceOverWiFiNetwork.host.wlan[0].mac (Ieee80211MacRWMA, id=26)
received self message: (cMessage)Timeout
state information: mode = DCF, state = WAITACK, backoff = 1, backoffPeriod = 0.00966, retryCounter =
processing event in state machine Ieee80211Mac State Machine
FSM Ieee80211Mac State Machine: leaving state  WAITACK
firing Transmit-Data-Failed transition for Ieee80211Mac State Machine
        IP Datagram with UDP payload
        Data for the new IP_NOTIFY packet:
        ID: 748
        Outcome: IP_NOTIFY_FAILURE
        Length (in byte): 1180
        More Fragment Field: 1
        Offset: 7080
        UDP Source Port: 1000
        UDP Destination Port: 2000
sending up (Ieee80211DataFrame)IPNotifyPacket-ID-748
dropping frame from transmission queue
requesting another frame from queue module
FSM Ieee80211Mac State Machine: entering state IDLE
FSM Ieee80211Mac State Machine: leaving state  IDLE
firing Immediate-Data-Ready transition for Ieee80211Mac State Machine
FSM Ieee80211Mac State Machine: entering state DEFER
FSM Ieee80211Mac State Machine: leaving state  DEFER
firing Immediate-Wait-DIFS transition for Ieee80211Mac State Machine
FSM Ieee80211Mac State Machine: entering state WAITDIFS
scheduling DIFS period
```

Figura 3.3: Creazione pacchetto IPNotify nel MAC

3.3.6 Monitor

Manca una ultima componente fondamentale affinché il meccanismo funzioni correttamente. Infatti l'ULB deve essere a conoscenza di quali e quante schede wireless sono attualmente configurate ed utilizzabili per inviare pacchetti.

Il monitor è stato implementato affinché segnali al livello applicazione quando una scheda viene correttamente associata ad un AP, o questa associazione viene meno. In caso di associazione o dissociazione, oppure in caso di perdita di troppi beacon, il monitor crea ed invia un messaggio di tipo ReconfNot (Reconfiguration Notification) all'ULB, notificando l'evento.

Il monitor è stato implementato nelle classi ieee80211MgmtSTARWMA

(.cc/h/ned)[8] e Ieee80211AgentSTARWMA (.cc/h/ned)[9]. I due moduli interagiscono tra loro per gestire tali eventi.

3.4 I pacchetti

Oltre i normali pacchetti UDPPacket, IPDatagram e Ieee80211DataFrame utilizzati per il trasporto dei dati, sono stati creati due ulteriori tipi di pacchetti per il trasporto delle notifiche all'interno del sistema RWMA.

3.4.1 IPNotify

Il pacchetto IPNotify è implementato nel file IPNotify.msg (da cui vengono generati automaticamente IPNotify_m.cc, IPNotify_m.h).

Viene utilizzato dalla componente TED per segnalare il riuscito o fallito tentativo di trasmissione di un frammento, e per ritornare l'identificatore univoco di un datagram IP appena creato.

La classe contiene i seguenti campi:

- identification (int, default -1)
- outcome enum(IPNotifyOption)
- length (int, default -1)
- morefragments (bool)
- fragmentOffset (int, default -1)

Dove l'enumerazione IPNotifyOption può assumere i seguenti valori interi:

- IP_NOTIFY_SUCCESS (0)
- IP_NOTIFY_FAILURE (1)

[8] Sottoclasse di ieee80211MgmtSTA (.cc/h/ned).
[9] Sottoclasse di Ieee80211AgentSTA (.cc/h/ned).

- IP_NOTIFY_RETURN_ID (2)

I primi due valori sono utilizzati direttamente dal TED, mentre il terzo segnala che il pacchetto ha il compito di ritornare un identificativo (nel qual caso i campi oltre identification e outcome non vengono utilizzati).

3.4.2 ReconfNot

Il pacchetto ReconfNot è implementato nel file RecNotification.msg (da cui vengono generati automaticamente RecNotification_m.cc, RecNotification_m.h).

Viene utilizzato dal Monitor per segnalare quando una interfaccia è correttamente configurata oppure non è più utilizzabile. Come il pacchetto IPNotify, è diretto all'ULB.

La classe contiene i seguenti campi:

- IPAddress interface
- int status enum(ReconfigurationNotificationOption)

Dove l'enumerazione ReconfigurationNotificationOption può assumere i seguenti valori interi:

- CONFIGURED (1)
- DISABLED (-1)

Capitolo 4

Progettazione

Tutto il codice è stato interamente progettato e strutturato per lavorare sul framework OMNeT++ [**?**] (v. 4.0). Il progetto si basa sull'implementazione del sistema RWMA realizzato da Fabrizio Sabatini [**?**] utilizzando la versione di INET modificata per supportare schede wireless multiple su ogni singola station, ad opera di Piero Murphy [**?**].

I moduli sono stati implementati come sottoclassi di classi preesistenti in INET aggiungendo, dove necessario, proprietà e comportamenti secondo le esigenze del progetto. Per questo si è preferito utilizzare, per quanto possibile, il meccanismo dell'ereditarietà in modo da mantenere le politiche di riusabilità ed estendibilità del progetto originale.

Infatti si è evitato di apportare modifiche ai comportamenti delle classi preesistenti, estendendole dove necessario, in modo che i nuovi sviluppi siano trasparenti rispetto all'utilizzo canonico.

L'implementazione dovrebbe risultare compatibile con versioni future di INET e OMNeT++, a meno di cambiamenti nelle interfacce delle classi.

4.1 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.

4.1.1 Infrastruttura

L'intera rete è formata da un insieme di sottoreti LAN (assimilabili anche a WAN e MAN), wired e wireless, interconnesse tra loro e raggruppate in due reti di classe B[2] collegate da due **backbone** con diversa latenze. In particolare uno dei due backbone presenta una latenza inaccettabile per una conversazione VoIP. La comunicazione tra host di reti diverse può avvenire grazie alla presenza di router che collegano due o più reti LAN ed inoltrano pacchetti utilizzando tabelle di routing.

4.1.2 Tipologia delle subnet

La tecnologia utilizzata per le reti wireless è la WiFi (IEEE 802.11) e ognuna di loro è costituita da un access point (AP) e da un insieme di nodi wireless che comunicano con altri nodi, così da generare **traffico di background** sui canali radio degli AP. In particolare si è implementata una rete wireless in cui le comunicazioni degli host generano un elevato tasso di corruzione nei dati trasmessi. Per quanto riguarda le reti su cavo (wired), queste utilizzano la tecnologia ethernet (IEEE 802.3) e sono costituite da un bus e

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

[2]Sono le reti i cui indirizzi IP usano i primi 2 byte come indirizzo di rete, e gli altri 2 byte come indirizzo per gli host.

da un insieme di nodi collegati ad esso. Anche nelle reti wired viene generato del traffico di background dai nodi presenti.

4.1.3 Indirizzi IP e subnet mask

Ad ogni rete e sottorete è stato assegnato un indirizzo di rete IP univoco e una subnet mask. Tutti gli host connessi a tale rete devono avere nel loro indirizzo IP, l'indirizzo di rete fisica in cui si trovano. Questo è importante poiché i router inoltrano sulla base degli indirizzi di rete.

Per ogni sottorete è stata scelta come subnet mask la `255.255.255.128`, quindi ogni subnet è in grado di contenere fino a un massimo di 128 host contemporaneamente.

4.2 Proxy Server SIP/RTP

Per superare firewall e sistemi NAT presenti nelle reti wireless e fare in modo che il nodo mobile sia sempre raggiungibile all'interno della rete, è stato implementato un **proxy server** fisso, esterno alla rete wireless, che fa da mediatore per il traffico dati tra i due end-sytem. Ciascun nodo mobile esegue un proxy client SIP/RTP che instaura un canale di comunicazione multi-percorso con il server utilizzando tutte le interfacce di rete disponibili (NIC).

4.2.1 Funzionamento

Ogni nodo wireless mobile è a conoscenza dell'indirizzo IP del proxy server esterno. Per comunicare con l'altro end-system, il nodo mobile trasmette i suoi pacchetti applicazione, di tipo RTP, al server, che a sua volta gli inoltra al destinatario effettivo. Essendo il server proxy basato sul protcollo SIP/RTP, esso mantiene una mappa [ID nodo, indirizzo IP] con la quale è in grado di individuare il destinatario esatto del messaggio. Le informazioni contenute nella mappa le ricava direttamente dal pacchetto RTP ogni volta che riceve

un nuovo messaggio da uno dei due end-system. Grazie a questo meccanismo la riconfigurazione di un nodo dovuta allo spostamento in una nuova rete, resta trasparente al secondo end-system, il quale continua a trasmettere i sui messaggi al server proxy. Come è già stato accennato, nella versione implementata il destinatario del nodo mobile è il server stesso.

4.3 Protocollo DHCP

Poiché gli host mobili, spostandosi, sono in grado di cambiare rete, è stato necessario implementare una versione semplificata del protocollo DHCP, assente nella libreria INET. Il protocollo DHCP è usato in Internet per consentire una configurazione automatica dei nodi IP che si collegano alla rete ed è eseguito sopra UDP.

La versione proposta differisce da quella originale in quanto, mentre nella prima il DHCP è un servizio di sistema, nella versione implementata è gestita direttamente a livello applicazione dal Load Balancer.

Grazie all'introduzione di questo protocollo nell'architettura RWMA, adesso il nodo mobile è in grado di ricevere pacchetti dal lato fisso, cosa che nella versione precedente del simulatore, sviluppata da Piero Murphy [**?**] e Fabrizio Sabatini [**?**], non accadeva a causa dell'irraggiungibilità del nodo.

4.3.1 Formato del pacchetto DHCP

Un pacchetto DHCP è formato dai seguenti campi principali:

- **type**: tipo del pacchetto DHCP (DHCPDISCOVER, DHCPREPLY);
- **srcMacAddr**: indirizzo MAC dell'interfaccia da configurare;
- **yiaddr**: indirizzo IP assegnato dal server al client;

4.3.2 Funzionamento

Un nodo che vuole configurare una propria interfaccia di rete, deve innanzitutto ricavare l'indirizzo di MAC dell'interfaccia, in modo da associare la richiesta all'interfaccia specificata. Successivamente inviare un pacchetto DHCP di tipo DHCPDISCOVER inserendovi l'indirizzo MAC dell'interfaccia in questione. La richiesta viene quindi trasmessa al server DHCP utilizzando il protocollo UDP. Il client, quindi, si comporta anche da agente di collegamento, che nell'implementazione reale è un nodo separato presente nella stessa rete locale in cui si trova il client e che si occupa di comunicare con il server DHCP attraverso una comunicazione unicast. Nell'implementazione si è scelto di accorpare l'agente di collegamento nel client per motivi prestazionali, essendo questo ininfluente ai fini della simulazione. Il server quando riceve un pacchetto DHCPDISCOVER, cerca un indirizzo disponibile tra quelli in suo possesso e se ne trova almeno uno, lo spedisce in risposta al client assieme all'indirizzo MAC dell'interfaccia mittente, inserendolo in un pacchetto DHCP di tipo DHCPREPLY. Il client ricevuto tale pacchetto, estrae le informazioni e assegna l'indirizzo IP all'interfaccia con indirizzo di MAC pari a quello presente nel pacchetto. Una volta configurata, l'interfaccia è raggiungibile dalla rete e può essere utilizzata dall'host per comunicare.

4.4 File

Per implementare il sistema sono stati creati o modificati diversi file. Verranno ora elencati a seconda del loro PATH (e di conseguenza per attinenza di livello OSI), segnalando con C quelli nuovi e con M quelli già esistenti che hanno subito modiche.

- src/applications/udpapp/ : DHCPPacket.msg/cc/h (C)
- src/applications/udpapp/ : ULBRWMAServerProxy.cc/h (C), ULBRWMA.cc/h (M)
- src/applications/udpapp/ : UDPBasicAppForServerProxy.cc/h (C)

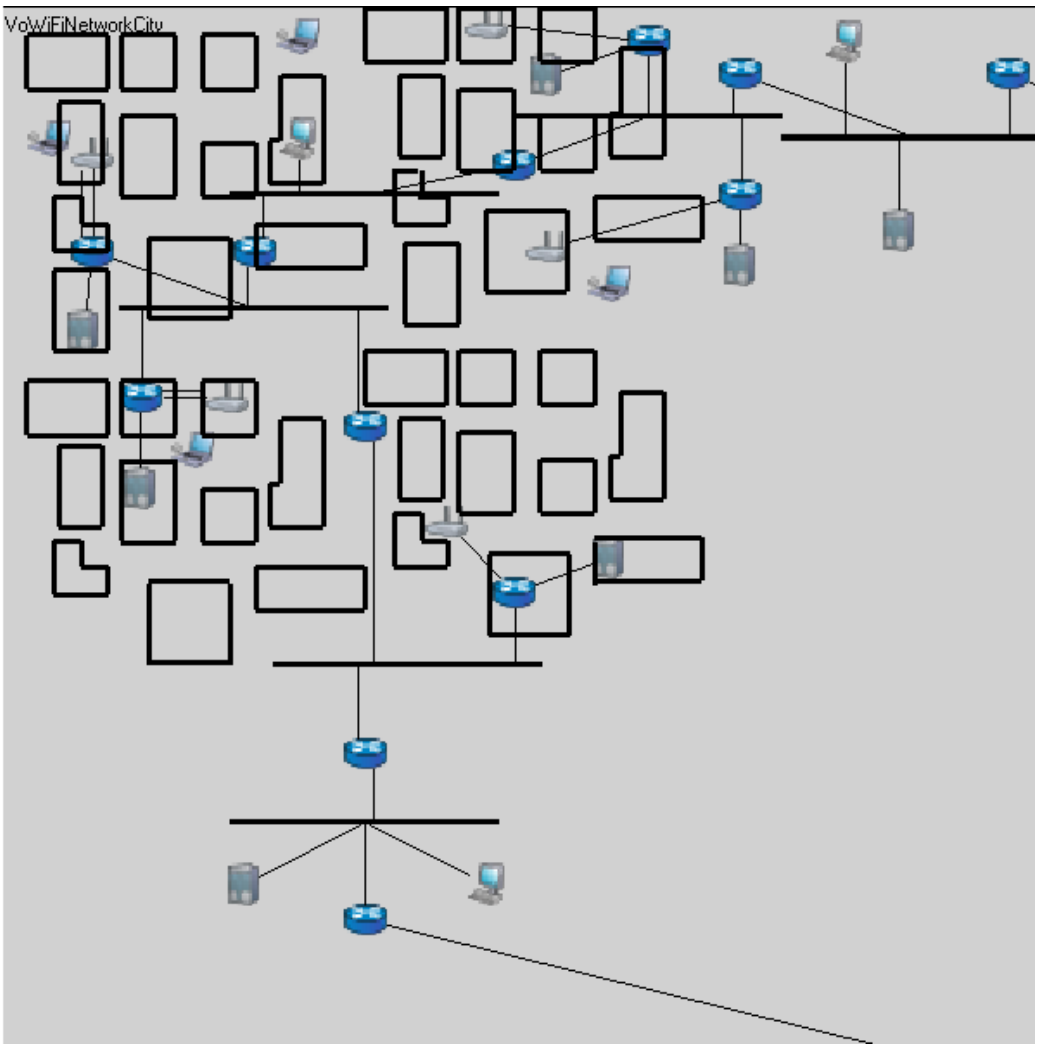

Figura 4.1: Rete di classe B del nodo mobile

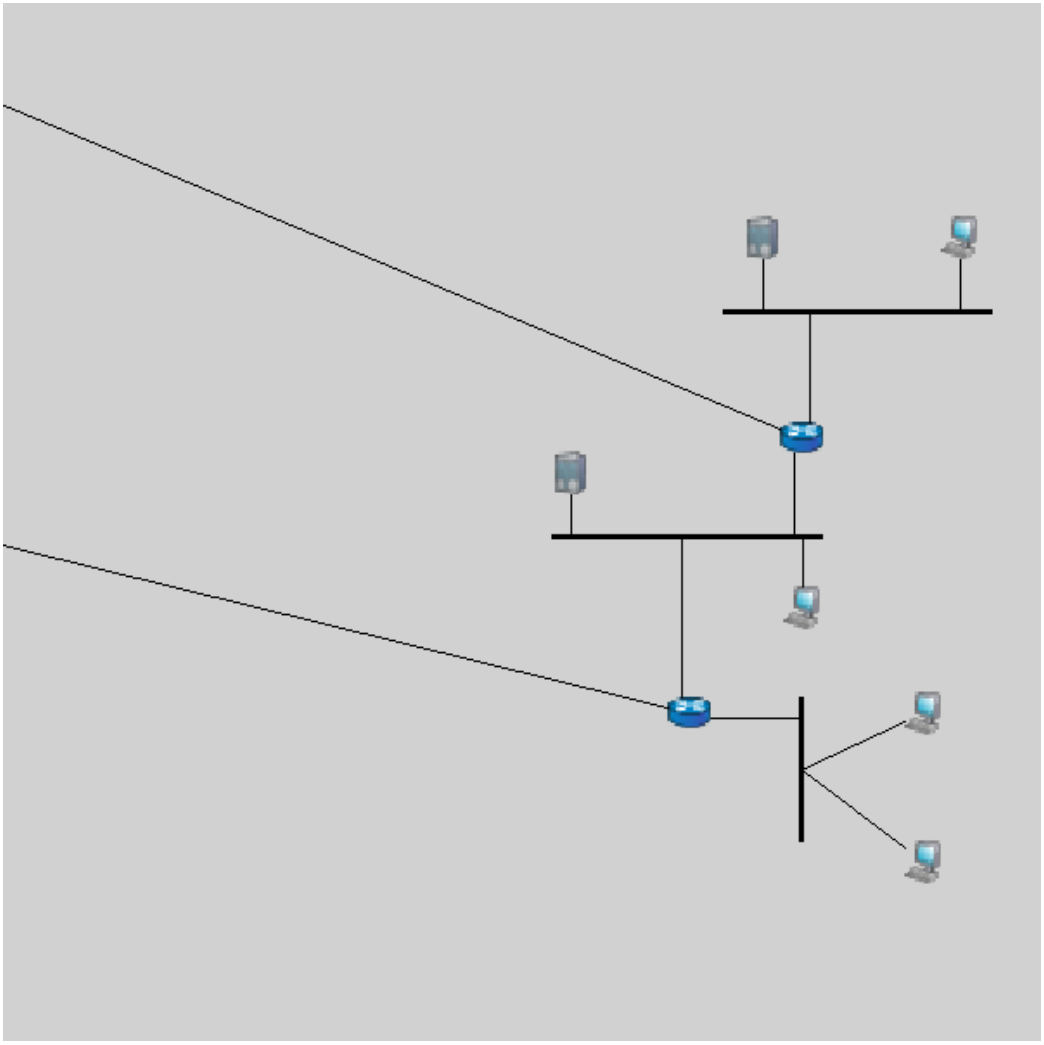

Figura 4.2: Rete di classe B del proxy server

Capitolo 5

Note implementative

L'implementazione proposta utilizza i moduli sviluppati da Piero Murphy [**?**] e Fabrizio Sabatini [**?**]. In particolare per quanto riguarda la comunicazione VoIP, sono stati utilizzati i due nodi già implementati da Fabrizio Sabatini. L'unica modifica è stata apportata al nodo proxy server per ottenere il comportamento descritto in (4.2). Anche per quanto riguarda la mobilità del nodo mobile, si è riutilizzata quella scritta da Sabatini.

La rete e i nodi vengono configurati tramite i file di configurazione VoWiFiNetworkCity.ned e rwmaMonitorCity.in.

5.1 Rete realistica

Per ottenere un comportamento realistico della rete, sono state riprodotte, all'interno di essa, situazione reali che si posso trovare oggi in Internet. Sono stati quindi implementati:

1. due backbone di diversa latenza per ricreare la circostanza in cui esistono più percorsi alternativi che collegano due end-system;

2. traffico di background nell'intera rete;

3. traffico wireless di background;

5.1.1 Backbone

I due backbone implementati sono rappresentati da due connessioni di tipo PPP (Point-to-Point Protocol) fra i router di bordo delle due reti B. Per ciascun backcbone 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.

5.1.2 Traffico di background

Il traffico di background viene generato da tutti gli host presenti nella rete, che comunicano tra di loro trasmettendo pacchetti a intervalli temporali prefissati. In particolare, considerando la rete come suddivisa in due grosse reti di classe B, ogni host di una rete comunica con un host dell'altra rete a eccezione di quelli wireless che si trovano tutti in un sola rete e comunicano solo tra di loro. Gli host fissi collegati via cavo, eseguono l'applicazione della libreria OMNeT++/INET denominata **UDPBasicApp** per comunicare, mentre quelli wireless, l'applicazione VoIP sviluppata da Piero Murphy [**?**] chiamata **UDPBasicAppForMultipleNics**.

5.1.3 Traffico di background wireless

Il traffico di background nei tratti wireless, viene generato da nodi wireless fissi posizionati in prossimità dell'access point della rete. In una di queste reti è presente un host che genera un flusso continuo di dati in modo da provocare interferenze al nodo mobile che si sposta in questa sottorete.

5.2 Proxy Server

Il proxy server è stato inserito in una rete diversa da quella del nodo mobile, poiché svolge anche la funzione di secondo end-sytem nella comunicazione VoIP.

Per ottenere il comportamento descritto in (4.2), è stato implementato un nuovo modulo **load balancer** dell'architettura RWMA, chiamato ULBRWMAServerProxy, che presenta le funzionalità di instradatore per i dati. Il modulo ULBRWMAServerProxy è utilizzato dall'applicazione proxy, denominata UDPBasicAppForServerProxy, che viene eseguita da un nodo fisso collegato via cavo alla rete. Il load balancer gestisce una mappa [ID, indirizzo IP] in cui inserisce l'ID e l' indirizzo IP dei client che si appoggiano al proxy server per comunicare. Ogni volta che il proxy server riceve un messaggio, il suo modulo load balancer, prima di passarlo all'applicazione proxy, invoca il metodo `updateAddrClient()`, che aggiorna le informazioni della mappa relative al client. Il metodo estrae ID e indirizzo IP del mittente. Se tale ID non è presente nella mappa, aggiunge una nuova entry (ID,indirizzo IP). Se invece è già presente, per mantenere l'informazione sull'indirizzo IP aggiornata, sovrascrive il vecchio indirizzo con quello appena letto. In questo modo i client sono sempre raggiungibili dal proxy server anche se cambiano indirizzo IP. Come avevamo già accennato il server si comporta anche da secondo end-system della comunicazione. L'applicazione UDPBasicAppForServerProxy che trasmette messaggi voce, in sostanza è la stessa dei client, ma si è dovuta creare una nuova classe poiché questa deve estendere quella del nuovo load balancer. L'applicazione genera messaggi che vengono instradati dal load balancer verso un destinatario scelto a caso fra quelli presenti nella mappa.

5.3 Protocollo DHCP

Il protocollo DHCP implementato, funziona solo nelle sottoreti wireless mentre nelle altre reti è assente poiché i nodi vengono configurati staticamente tramite file .ini. Ogni rete wireless ha un nodo che fa da server DHCP per la sottorete. Il server è configurato, tramite file .ini, con l'indirizzo IP `128.128.128.128` (un indirizzo noto ai client) e un set di indirizzi IP relativo alla sottorete in cui si trova, cioè che hanno come indirizzo di rete quello

della subnet. I nodi che vogliono configurare una propria interfaccia, dopo aver trasmesso il messaggio DHCPDISCOVER, cambiano temporaneamente l'indirizzo dell'interfaccia da configurare e dalla quale riceveranno la risposta del server. Questo è necessario affinché il client possa ricevere la risposta. Infatti un nodo wireless spostandosi in un nuova subnet diventa irraggiungibile dalla rete a causa dell'indirizzo IP non conforme alla nuova sottorete. L'indirizzo temporaneo che usa è `64.64.64.64` che è noto al server. Il server, quindi, quando risponde, trasmette il messaggio DHCPREPLY all'indirizzo `64.64.64.64`. Ottenuto il nuovo indirizzo, il client lo assegna all'interfaccia che diventa così raggiungibile dalla rete.

Capitolo 6

Test

In questo capitolo verrà spiegato come poter compilare il codice ed eseguire le simulazioni presenti. Prerequisito necessario affinché le operazioni che verranno introdotte vadano a buon fine, è avere sul proprio sistema il framework OMNeT++ correttamente compilato e configurato.

6.1 Compilare il codice

Per compilare il codice (ad esempio a seguito di una modifica dei file), bisogna invocare lo script **makeandcopy**, che si occupa di compilare ed infine copiare l'eseguibile nella cartella di destinazione.

Se, invece, si creano dei nuovi file, il makefile va aggiornato lanciando il comando **make -f makemakefiles** e poi invocare lo script precedente per avere il sistema funzionante.

Un possibile problema che si può riscontrare nel lancio del comando sopra citato, è dato dalla presenza della documentazione del codice all'interno della cartella **doc**. Nel caso il comando fallisca, si può spostare la documentazione fuori dalla cartella principale e riprovare nuovamente.

Se ad essere modificati sono solo file ned (come nel caso si intenda usare un modulo al posto di un'altro) o file ini, non è necessario ricompilare, ma basta lanciare nuovamente la simulazione.

6.2 Simulazioni

La simulazione creata per testare il sistema, si trova nella cartella /examples/-wireless/voiceoverwifiRWMA, nel file ini: `rwmaMonitorCity.ini`

Per la simulazione del'ambiente urbano è stata creata una configurazione nel file `rwmaMonitorCity.ini` chiamata **NetworkCity**, che fa riferimento alla **network** definita nel file `VoWiFiNetworkCity.ned`.

Per facilitarne il lancio, è stato creato, all'interno della cartella, uno script e che evita all'utente di dover specificare il percorso dell'eseguibile di INET. Quindi, per eseguire una simulazione, basta lanciare il comando:

`./run <inifile.ini>`

Eseguendo la simulazione col comando `./run rwmaMonitorCity.ini` viene mostrata una finestra contenente la configurazione **NetworkCity**:

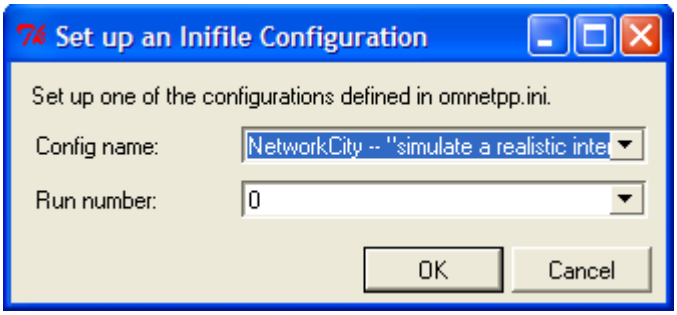

Figura 6.1: Menù di scelta della simulazione

6.2.1 La simulazione NetworkCity

Lo scenario **NetworkCity** è costituito dalle due reti in cui sono presenti i vari hosti. Il dispositivo wireless mobile, si muove lungo un percorso predefinito, passando tra un certo numero di ostacoli disposti così da rappresentare il perimetro di edifici in un contesto urbano. Nella rete dell'host mobile sono inoltre presenti 5 Access Point collocati in modo tale da fornire una copertura disomogenea dello scenario per far sì che l'host mobile passi per aree con intenstità di segnale diversa.

Nella simulazione si mostra il comportamento della rete costruita in presenza di più host comunicanti. Inoltre si testa il funzionamente del proxy server relativamente al suo ruolo di instradatore e del protocollo DHCP.

La configurazione utilizzata è costituita da:

- Periodo di tempo simulato: $3600s$
- Canali
  - AP: statico, scelto con distribuzione uniforme nell'intervallo $[1, 11]$
  - Host mobile: scansiona tutti i canali tra 1 e 11
- Host Mobile:
  - Applicazione: **UDPBasicAppForMultipleNics**
  - Dimensione dei messaggi: $100B$
  - Frequenza dei messaggi: $40ms$
  - Indirizzo IP temporaneo: 64.64.64.64
  - Indirizzo IP del server DHCP: 128.128.128.128
- Server Proxy:
  - Applicazione: **UDPBasicAppForServerProxy**
  - Dimensione dei messaggi: $100B$
  - Frequenza dei messaggi: $40ms$
- Server DHCP:
  - Applicazione: **UDPBasicAppForServerProxy**
  - Indirizzo IP del client: 64.64.64.64
  - Indirizzi IP per i client DHCP (sottorete wireless 1):

    `128.96.4.132 128.96.4.133 128.96.4.134 128.96.4.135 128.96.4.136`

  - Indirizzi IP per i client DHCP (sottorete wireless 2):

    `128.96.4.4 128.96.4.5 128.96.4.6 128.96.4.7 128.96.4.8`

- Indirizzi IP per i client DHCP (sottorete wireless 3):

`128.96.6.3 128.96.6.4 128.96.6.5 128.96.6.6 128.96.6.7`

- Indirizzi IP per i client DHCP (sottorete wireless 4):

`128.96.5.6 128.96.5.7 128.96.5.8 128.96.5.9 128.96.5.10`

- Indirizzi IP per i client DHCP (sottorete wireless 5):

`128.96.5.132 128.96.5.133 128.96.5.134 128.96.5.135 128.96.5.136`

- Mobilità

  - Modulo: **TurtleMobility**

  - Percorso: descritto nel file `path.xml`

  - Velocità: scelta con distribuzione uniforme nell'intervallo $[1m/s, 2m/s]$ per ogni segmento del percorso

6.2.2 Alcuni risultati

Sebbene un' analisi accurata della simulazione richederebbe l'esecuzione di parecchi run tra i quali calcolare valori medi e intervalli di confidenza, si espongono in questo paragrafo i risultati di un esecuzione relativi al proxy server e al protocollo DHCP. Nelle prime due immagini si mostra il numero di pacchetti ricevuti e inviati dal nodo mobile e dal proxy server/secondo end-system, che dimostra il corretto funzionamento del server. Mentre per quanto riguarda il protocollo DHCP, nella terza e quarta figura viene mostrato il cambiamento di indirizzo IP del nodo mobile dopo essere entrato in una nuova sottorete.

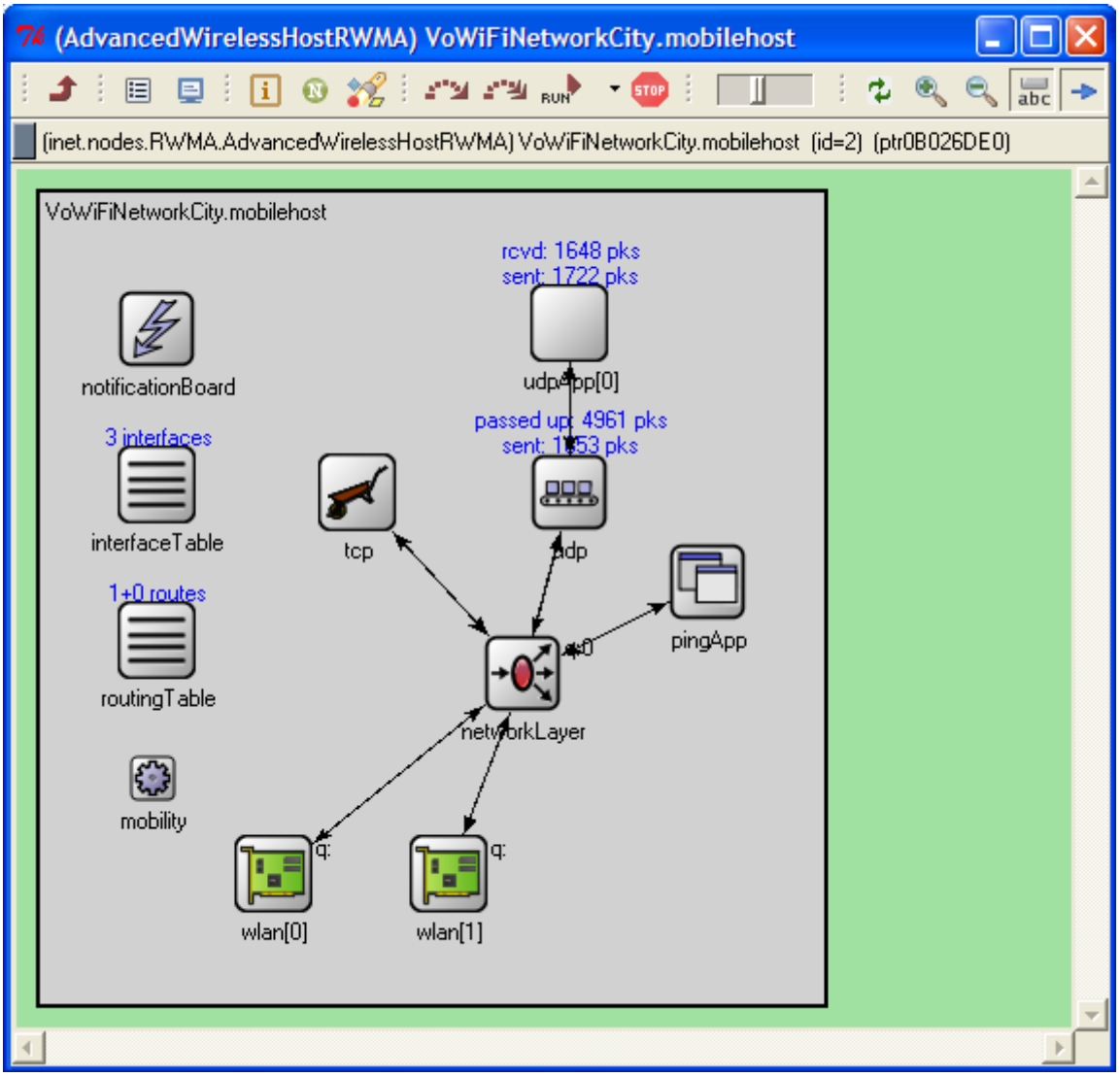

Figura 6.2: Messaggi ricevuti e inviati dal mobile

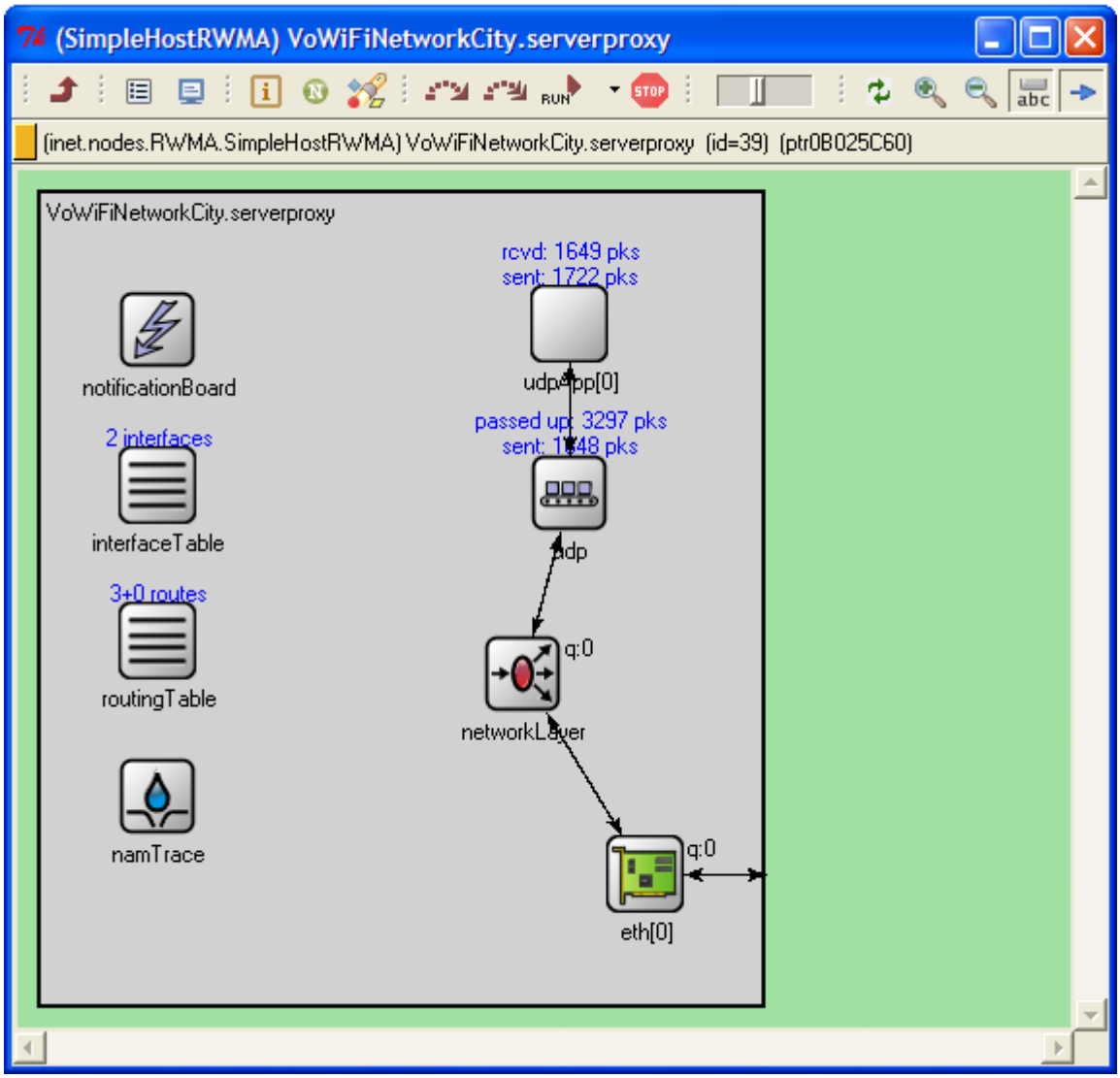

Figura 6.3: Messaggi ricevuti e inviati dal server proxy/secondo end-system

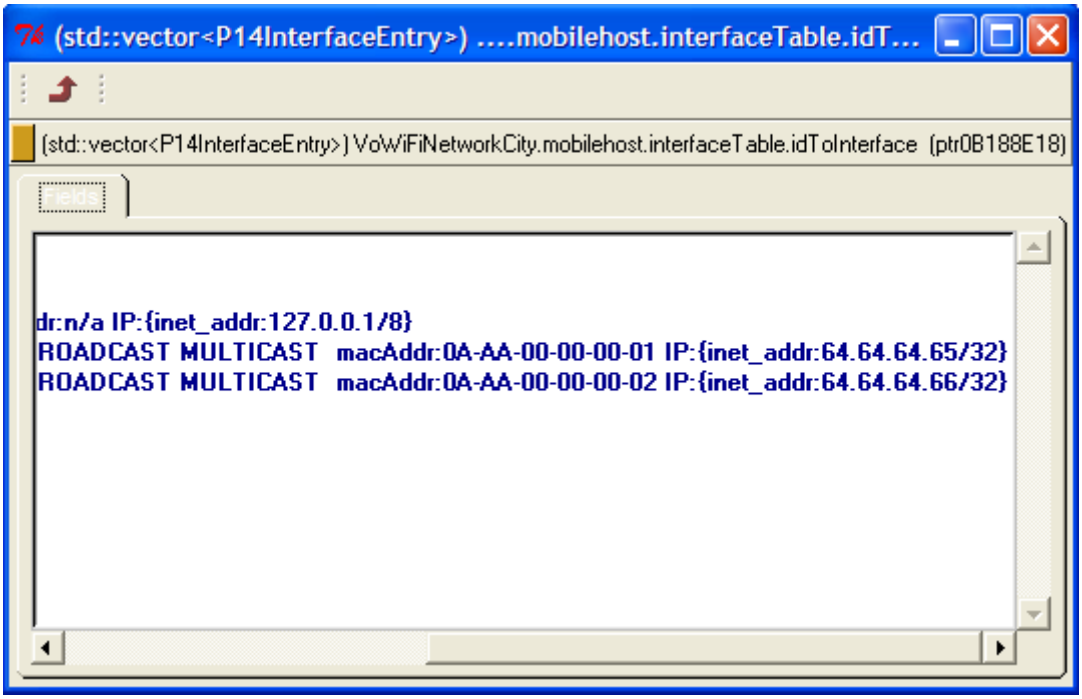

Figura 6.4: Interfacce del mobile all'nizio della simulazione

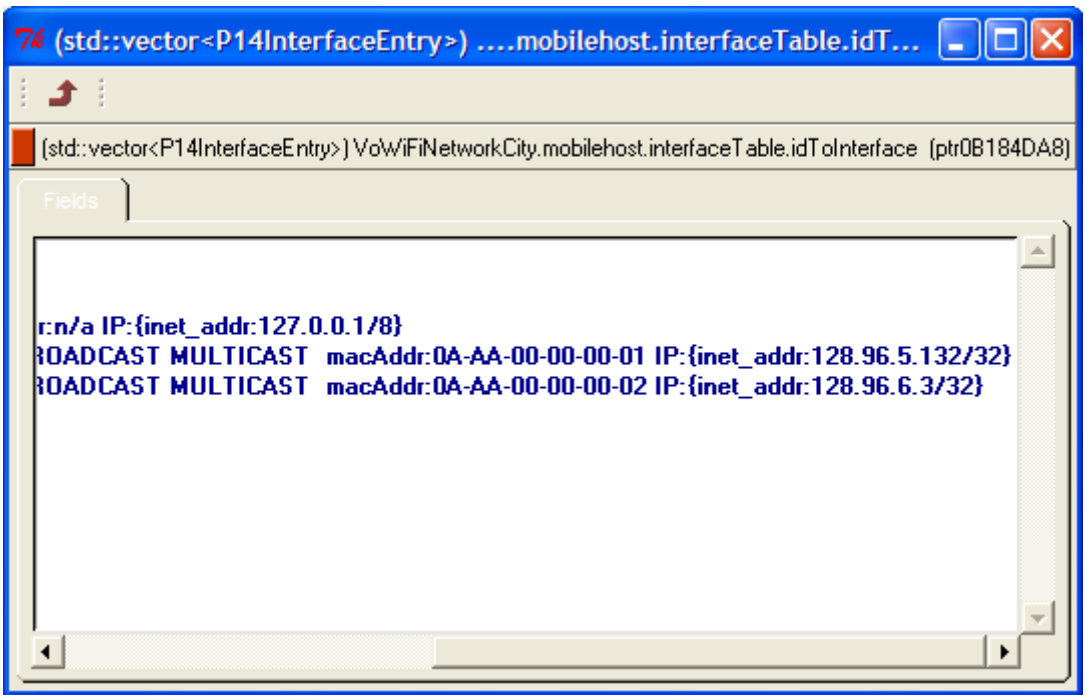

Figura 6.5: Situazione delle interfacce dopo circa 45min di esecuzione

Capitolo 7

Conclusioni

Grazie all'implementazione del proxy server e del protocollo DHCP si è potuto finalmente iniziare a testare le reali prestazioni dell'architettura RWMA su uno scenario di rete cittadina.

Il comportamento dei moduli sviluppati si è dimostrato essere come ce lo si aspettava. In particolare il protocollo DHCP funziona correttamente in presenza di host che si vogliono configurare. L'unico inconveniente potrebbe derivare dall'assenza di controllo sull'effettiva ricezione del pacchetto DHCP, sia da parte del server che del clent, evenienza possibile data la comunicazione wireless. Mentre per quanto riguarda la rete implementata, il suo comportamento si è dimostrato essere pienamente soddisfacente. Inoltre costituisce solo un'esempio semplificato di Internet, è perciò possibile implementare reti molto più complesse in modo da eseguire test sempre più affidabili.

Capitolo 8

Sviluppi futuri

8.1 Refactoring e compatibilità con INET

L'implementazione del supporto per NIC multipli risulta, attualmente, poco riutilizzabile ed incompatibile con la linea di sviluppo principale di INET, in quanto non segue un modello object oriented. Per questo sarebbe opportuno riprogettare il meccanismo in modo da estendere le classi preesistenti, senza necessariamente apportarvi modifiche.

In particolare sarebbe opportuno ripristinare i moduli ChannelControl e AbstractRadio originali e crearne, quindi, sottoclassi ed, inoltre, reimplementare la struttua `HostRef` come classe, utilizzata per identificare gli host wireless.

8.2 BasicObstacle

Allo scopo di rendere maggiormente realistico il modello degli ostacoli e di semplificarne la gestione da parte del ChannelControl, lo si potrebbe pensare come un caso particolare di host wireless il quale ritrasmetterebbe ogni **AirFrame** ricevuto con una minore potenza del segnale. Sarebbe necessario, però, gestire la duplicazione degli **AirFrame** che ne deriverebbe ed una dif-

ferente intensità di segnale nelle diverse direzioni[1]. Una possibilità sarebbe quella di gestire la propagazione del segnale con algoritmi di *ray tracing*[**?**].

8.3 ARPRWMA

Allo scopo di snellire l'implementazione e di aumentarne il realismo, si potrebbe incapsulare in un *datagram IP* la notifica di riconfigurazione dell'interfaccia wireless, attualmente identificata da `ReconfNot`, prima di inoltrarla al modulo ARPRWMA, il quale dovrebbe gestirla di conseguenza.

8.4 Canali per NIC multipli

Come accennato nel paragrafo **??**, nell'attuale implementazione le caratteristiche di ciascun NIC appartenenete ad un host wireless vengono mantenute in un contenitore di tipo `std::map` che non permette indici duplicati ed in questo modo non è possibile che due interfacce si configurino sullo stesso canale. Una possibile soluzione sarebbe quella di utilizzare un altro tipo di struttura dati, stando comunque attenti a non aumentare eccessivamente la complessità degli accessi.

8.5 Secondo end-system della comunicazione

Dato che si vorrebbe simulare un situazione reale in cui due nodi comunicano tramite proxy server, sarebbe bene introdurre il secondo end-system nella rete, al momento emulato dal proxy server stesso, in modo da effettuare test più affidabili. L'introduzione del secondo end-sytem è immediata e non comporta modifiche ai moduli eistenti. Basta infatti solo configurare il secondo nodo in modo che comunichi con il server.

[1] L'attenuazione del segnale dovuta alla riflessione ed all'oscuramento sono differenti

8.6 Protocollo DHCP

Come già stato accennato, l'implementazione data del protocollo DHCP è solo una semplificazione di quella originale. Sarebbe, quindi, bene modificarlo in modo che rispetti le specifiche reali del protocollo. In particolare l'ideale sarebbe quello di renderlo come servizio (applicazione) indipendente dalle altre applicazioni in esecuzione nel nodo.

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. 802.11n, “Higher Throughput Improvements using MIMO”, IEEE Standard for Information Technology, 2007

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

[8] ANSI/IEEE Std 802.11, 1999 Edition

[9] OMNet++ staff, OMNet++ v4.0 User Manual, Website, <http://www.omnetpp.org/doc/manual/usman.html>

[10] OMNet++ v4.0, <http://www.omnetpp.org/omnetpp>

[11] INET staff, INET Framework Model Documentation, Websiste, <http://inet.omnetpp.org/doc/INET/neddoc/index.html>

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

[13] Fabrizio Sabatini, "Un simulatore per un protocollo di QoS per reti wireless", Università degli Studi di Bologna, Non pubblicato

[14] Marco Pattoro e Francesca Montevecchi, "Scenario simulativo pe applicazioni VoIP su WiFi", Università degli Studi di Bologna, Non pubblicato

[15] Kari Laasonen, "Radio Propagation Modeling", Website, 2003, <http://www.cs.helsinki.fi/u/floreen/adhoc/laasonen.pdf>

[16] Doxygen 1.5.8, <http://www.doxygen.org/>

[17] 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

[18] Mohamed AYADI, Sami TABBANE, Ziad BELHADJ, "RAY-TRACING MODEL FOR URBAN MICRO-CELLULAR SYSTEM AT 1800 MHZ BAND", Ecole Supérieure des Communications de Tunis (Sup'Com), 2005