Alma Mater Studiorum · Università di Bologna

Scuola di Scienze
Corso di Laurea in Informatica

Simulazione VoIP di molteplici nodi mobili con percorsi casuali

Tesi di Laurea in Reti di Calcolatori

Relatore:
Dott.
Vittorio Ghini

Presentata da:
Mauro Licini

Sessione II
Anno Accademico 2012/2013

Introduzione

Negli ultimi anni, grazie alla sempre più rapida evoluzione e diffusione di dispositivi mobili, come *smartphone* e *netbook*, insieme alla maggior diffusione di reti **wireless** pubbliche nei centri urbani e punti di accesso (**AP**), 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, soprattutto in caso di mobilità su scala urbana. Per mantenere un livello di **Quality of Service (QoS)** accettabile per applicazioni di telefonia in questo tipo di scenari, è stato precedentemente realizzato il meccanismo **Robust Wireless Medium Access (RWMA)**[10] nel kernel del sistema operativo open source GNU/Linux. È stata sviluppata una simulazione per il meccanismo RWMA che tenga conto delle condizioni di applicabilità, delle prestazioni e dell'affidabilità.

Questa tesi si occupa principalmente della simulazione di più nodi mobili all'interno di uno scenario urbano su OmNet++. A questo scopo, è stato perfezionato un modello preesistente, precedentemente ampliato da Marco Ciamarella, dove era stato definito uno scenario simulativo rappresentante una stazione WiFi (host) mobile che si spostava tra le vie di un ipotetico centro cittadino in presenza di vari AP eseguendo un'applicazione VoIP grazie al meccanismo RWMA.

Nello specifico, nel **Capitolo 1** verrà fatta una panoramica sulle caratteristiche delle reti wireless IEEE802.11 e i vari protocolli di VoIP.

Nel **Capitolo 2** verrà descritto il meccanismo RWMA.

Il **Capitolo 3** presenterà il sistema simulativo OmNet++ e il framework INET, utilizzati nello sviluppo del progetto.

Nel **Capitolo 4** e **5** verranno illustrati gli obiettivi della tesi e i dettagli progettuali e implementativi della simulazione realizzata.

Infine, il **Capitolo 6** spiegherà come eseguire la simulazione.

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 uno specifico 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

L'installazione di reti wireless può avvenire in due diverse modalità:

Infrastructure Basic Service Set Questa infrastruttura, usata in **questa tesi** per le simulazioni, consiste in un insieme di **nodi** (Station e Wireless 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).

[1] Wireless Local Area Network. 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. Il termine indica dispositivi che possono collegarsi a reti locali senza fili (WLAN).

Indipendent Basic Service Set Una WLAN IBSS, 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 e dei 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: apparato che esegue l'interconnessione di reti locali multiprotocollo e gestisce l'instradamento dei messaggi attraverso due reti.

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

3. Client

1.1.2 Terminologia

Hotspot Sono così definiti gli Access Point pubblici.

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

Station (STA) 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[3].

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

Lavora sui livelli OSI[4] 1 e 2 ed è costruita secondo lo standard IEEE 802.11.
Alcuni parametri tipici di una scheda wireless sono:

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

Una scheda wireless può operare in due modalità:

1. Infrastructure
2. Ad-Hoc

Nella prima modalità, la scheda ha bisogno di un AP come intermediario e tutti i dati (di tutte le station) sono trasferiti usandolo come nodo di interconnessione. Per connettersi ad SSID[5] 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.

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

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

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.

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

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

IEEE 802.11 b Questo protocollo è stato ratificato nell'Ottobre del 1999; utilizza il CSMA/CA[6] come metodo di trasmissione delle informazioni e ha una capacità massima di 11Mb/s, un throughput[7] di circa 5 Mb/s e una portata massima all'esterno (in assenza di ostacoli) di circa 90-100 metri[3].

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

IEEE 802.11 n La versione definitiva dello standard è stata approvata l'11 Settembre 2009[5] e la sua pubblicazione è avvenuta il 29 ottobre 2009. Compatibile con il b, il protocollo è 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:

- Data Frame: usati per la trasmissione e il trasporto vero e proprio dei dati
- Management Frame: servono a scambiarsi informazioni

[6] Carrier Sense Multiple Access with Collision Avoidance (accesso multiplo con rilevazione di portante ed eliminazione di collisione) è un protocollo di trasmissione che evita le contese, cioè i tentativi di più stazioni di accedere contemporaneamente alla rete.

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

- Control Frame: gestiscono la trasmissione dei dati

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), 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: Data, Management, Control

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[8] - Inizializzato a 1 se il frame proviene dal sistema di distribuzione

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

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

8. Pwr mgt (Risparmio energia) - 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. More Data (Altri dati) - 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[9]

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

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

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

Per una migliore comprensione del lavoro svolto, ci soffermeremo maggiormente sui frame di gestione utilizzati per la configurazione, il mantenimento e il rilascio del canale di comunicazione e sui frame di controllo, pittusto che sui frame per il trasporto dei dati.

1.2.3 Management Frame

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, 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 a un AP quando desidera terminare una comunicazione; l'AP libera la memoria allocata e rimuove la scheda dalla tabella delle associazioni.

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

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

Disassociation frame Viene inviato se si vuole terminare una associazione.

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

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

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

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

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

1.2.4 Control Frame

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)

ACK Dopo aver ricevuto un frame dati senza errori, l'AP invia un frame ACK al WNIC del mittente. Se 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.

RTS e CTS Questi due tipi di frame implementano un meccanismo per ridurre le collisioni tra AP e station nascoste. Un 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.

1.3 Protocolli

1.3.1 Voice Over Internet Protocol

Il lavoro svolto si basa su 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 a costo zero, se si esclude il costo della connessione, con chiunque sia collegato alla stessa rete ed abbia una applicazione compatibile con la nostra.

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, invece, 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[10] è la scelta più comune per questo tipo di utilizzo; trattandosi di un protocollo di livello applicazione, il mittente ed il destinatario sono client VoIP.

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

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

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

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

I pacchetti vengono infatti accettati secondo l'ordine di arrivo e non quello di invio e 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.

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

1.3.3 Session Initiation Protocol

Il protocollo SIP è un protocollo del livello applicazione. Pur essendo stato progettato per applicazioni di tipo diverso e con potenzialità differenti da HTTP, è simile ad esso, in quanto è basato su un modello richiesta/risposta. 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: spettro di funzioni che comprendono il trasferimento di sessioni (per realizzare ad esempio il call forwarding, trasferimento di chiamata) e la modifica dei parametri di sessione

1.3.4 User Datagram Protocol

Lo User Datagram Protocol[11][1] (UDP) è un protocollo di livello trasporto connectionless[12] e stateless.

La comunicazione è implementata trasmettendo i pacchetti, chiamati Datagram[13], dal mittente al destinatario senza verificare lo stato stato della rete ne quello del ricevente.

Infatti, UDP non gestisce nessun tipo di controllo di flusso, nè 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 ed inoltre 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).

[11] Il protocollo e' stato definito nel 1980 da David P. Reed (RFC 768).

[12] I protocolli connectionless sono caratterizzati dalla particolarità di non configurare una connessione end-to-end dedicata tra due nodi prima di un invio.

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

Il protocollo UDP è quindi:

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

Gli unici servizi che UDP implementa sono il multiplexing (grazie alle porte) e il controllo dell'integrità (grazie al checksum), mentre 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 quali 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[14][2] (IP), definito nell'Internet Protocol Suite è il protocollo primario delle comunicazioni di rete e ha il compito di trasportare pacchetti dal mittente al destinatario basandosi solamente sul loro indirizzo.

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

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

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

È un protocollo di tipo best-effort, infatti non garantisce la consegna dei pacchetti a destinazione nè 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 (ad esempio 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[15] attraverso il quale il pacchetto può passare prima di essere scartato
11. Protocollo (8 bit): TCP, UDP, ICMP, ecc...

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

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[16] 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.

[16]Maximum transmission unit è la dimensione massima che può avare un pacchetto per viaggiare su una data rete.

Capitolo 2

Robust Wireless Medium Access

Nei lavori svolti precedentemente era stata ricreata, con simulatore OmNent++, una comunicazione VoIP tra un end-sytem wireless mobile e un altro fisso o mobile situati in due reti diverse. I due end-system operanti in uno scenario cittadino, utilizzavano un'architettura denominata RWMA(Robust Wireless Medium Access) che forniva 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.

2.1 Modello QoS

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

[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.

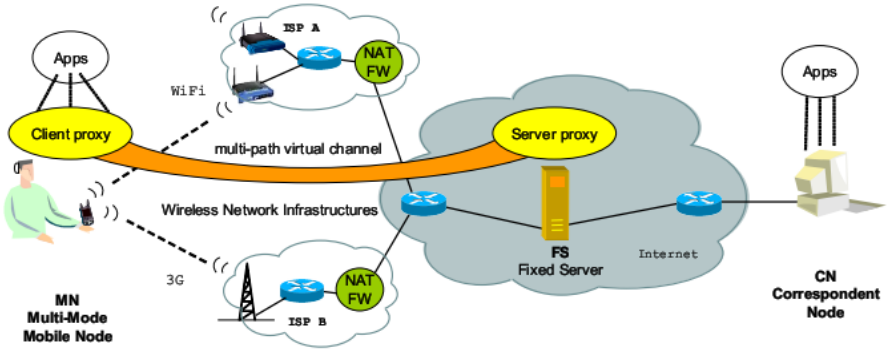

Figura 2.1: Possibile scenario di utilizzo

2. 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 di 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, di trasmetterli (e se necessario ritrasmetterli) al destinatario.

Il sistema implementato nella station è formato da tre componenti:

1. Transmission Error Detector (TED)

2. UDP Load Balancer (ULB)

3. Monitor

L'implementazione nel server riceve i datagram, li riordina e li passa all'applicazione per la riproduzione. È importante che il server che riceve il messaggio

sia a conoscenza di quale interfaccia è stata usata per inviare pacchetto (al fine di rispondere correttamente).

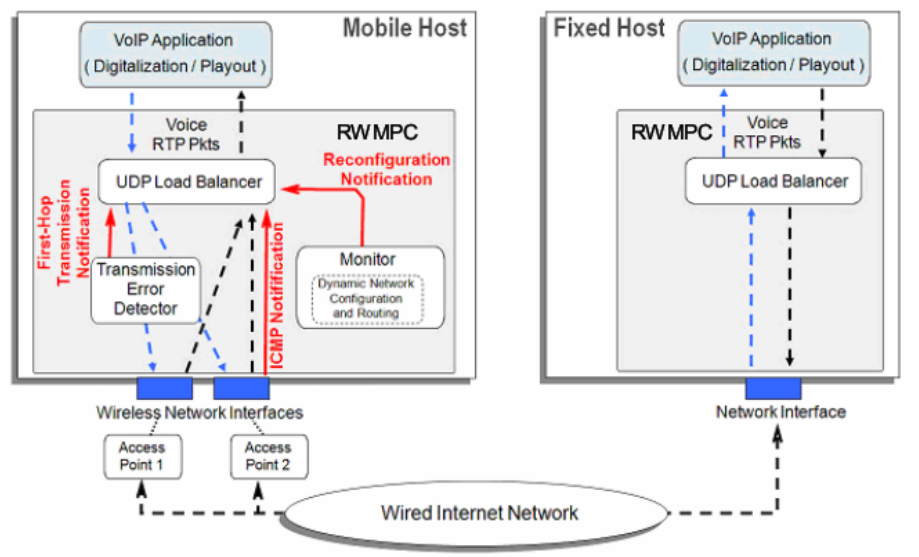

Figura 2.2: Architettura del sistema RWMA

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[3, 4].

Si occupa di controllare se ogni singolo datagram UDP è stato ricevuto con successo, attraverso un collegamento configurato e attivo, dall'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".

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]).

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

Questo permette di implementare un semplice meccanismo per il controllo del flusso della trasmissione a basso livello per pacchetti UDP, che altrimenti non godrebbe di nessun tipo di controllo simile.

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.

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, di incapsularli in datagram UDP e di inviarli al destinatario. Attraverso le notifiche di errore ricevute da più fonti, 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 i quali 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, il Monitor si occupa di notificare all'ULB 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.

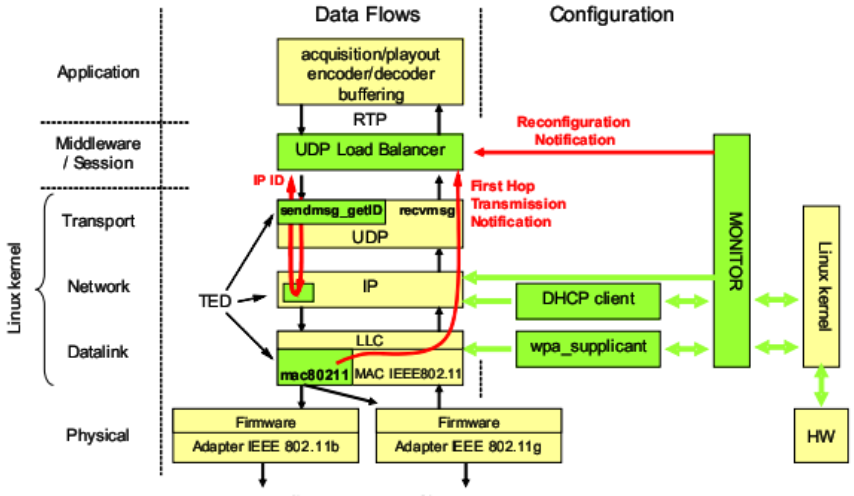

Figura 2.3: Implementazione RWMA su Linux

[3]La Routing Table 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.

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).

[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.

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.

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:

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

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

2.2.4 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.

2.2.5 Server DHCP

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. Il server DHCP gestisce l'assegnazione degli indirizzi IP in modo dinamico , ovvero quando un client richiede un IP questo non gli viene assegnato in modo permanente ma soltanto per un certo periodo di tempo , al termine del quale l'assegnazione deve essere rinnovata: il client può vedersi riassegnare lo stesso IP oppure (nel caso in cui questo non sia più disponibile) il server DHCP gliene assegnerà un altro. Essendo necessario, per le nostre simulazioni, un server DHCP che permetta ai nodi mobili di cambiare indirizzo IP ogni volta che si spostano in una sottorete diversa, il protocollo DHCP è stato implementato in modo che sia noto, all'host che si vuole configurare, l'indirizzo IP del server DHCP e che non avvenga alcuna trasmissione broadcast delle richieste DHCP. Si ha, quindi, che l'host funziona anche da agente di collegamento[7].

[7]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.

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 come funziona il simulatore di reti open source OMNeT++[6] e il relativo framework INET[8], su cui si basa l'intera implementazione.

3.1 OMNeT++

OMNeT++ è un software modulare per lo sviluppo di simulazioni ad eventi discreti. Non è propriamente un simulatore, ma piuttosto un'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

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

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[7].

3.1.1 Moduli

Le componenti fondamentali del software sono delle classi riutilizzabili chiamate **moduli**, realizzate 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** e rappresentano delle entità e/o dei comportamenti. Il modulo di livello massimo viene invece chiamato **system module** e racchiude tutto il sistema nel suo complesso.

I moduli semplici vengono assemblati in componenti più grandi e complessi chiamati **compound modules** o in **modelli** (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 ad altri 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 sottomoduli, gate o connessioni interne.

I parametri possono essere assegnati nei file NED, nei file di configurazione (con estensione .ini), 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** 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. I messaggi hanno attributi standard (timestamp) e possono arrivare da un altro modulo oppure dal modulo stesso, nel qual caso vengono definiti **self-message** il cui invio è gestito con dei timer.

Il tempo locale di simulazione di un modulo è strettamente legato ai messaggi e 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.

Quando, invece, 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 che contiene informazioni ausiliarie necessarie al livello a cui è diretto, ma che non sono da inoltrare ad altri moduli. Gli oggetti control info possono ovviamente essere definiti dall'utente, e sono sottoclassi di cObject (libreria OMNeT++).

3.1.5 Il linguaggio NED

Il termine NED fa riferimento a **Network Description** e si tratta di un linguaggio di alto livello usato dall'utente per descrivere la struttura di un modello.

Questo linguaggio 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 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 e 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 da un punto (dal modulo di massimo livello al modulo contenente il parametro).

I parametri vengono assegnati attraverso l'operatore di uguaglianza matematica (=) 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

3.2 INET

Il framework INET è costruito 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 e 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, come router ed altri dispositivi di rete. Tra questi segnaliamo:

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

Un'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)

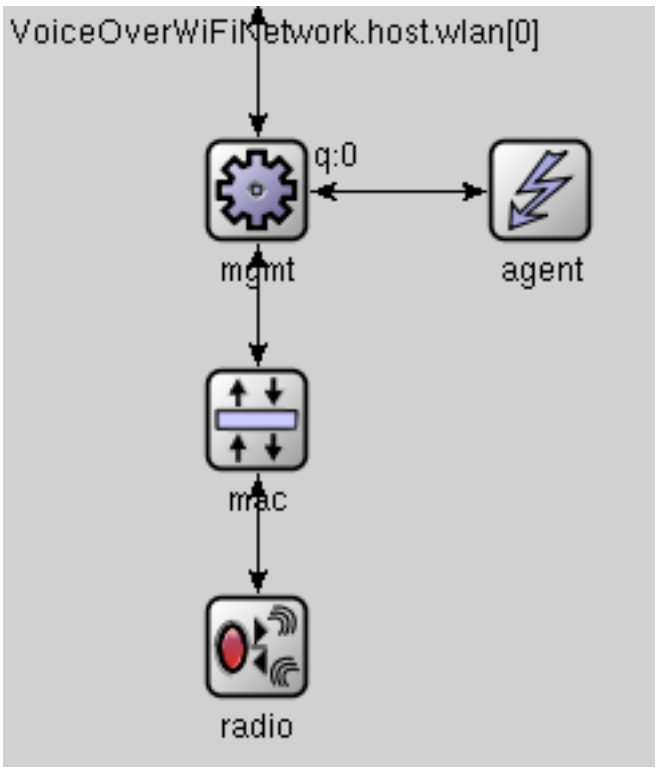

Figura 3.1: Architettura di una WNIC in INET

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

Il manager Si occupa di incapsulare e decapsulare i messaggi per (o dal) livello MAC e scambiare frame di gestione con altre station o Access Point. 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. Proprio come l'agent, ha differenti implementazioni a seconda del suo ruolo.

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 Si occupa della trasmissione e ricezione dei frame, modella le caratteristiche del canale radio e determina se un frame è stato ricevuto correttamente o no (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)
- inet_addr: indirizzo IP
- Mask: netmask
- Groups: gruppo Multicast
- MTU: MTU del canale
- Metric
- flag: BROADCAST, MULTICAST, POINTTOPOINT

[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.

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 due diverse modalità: attraverso l'interfaccia grafica (default), comoda per le dimostrazioni e il debugging e la versione da linea di comando, sicuramente più adatta per esecuzioni batch.

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

Per eseguire una simulazione 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*>
- /PATH_TO_INET/run_inet -c <*nomeDellaSimulazione fileDiConfigurazione.ini*>

Il primo 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 il secondo tipo si usa nel caso si voglia lanciare direttamente una simulazione specifica all'interno del file.

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.

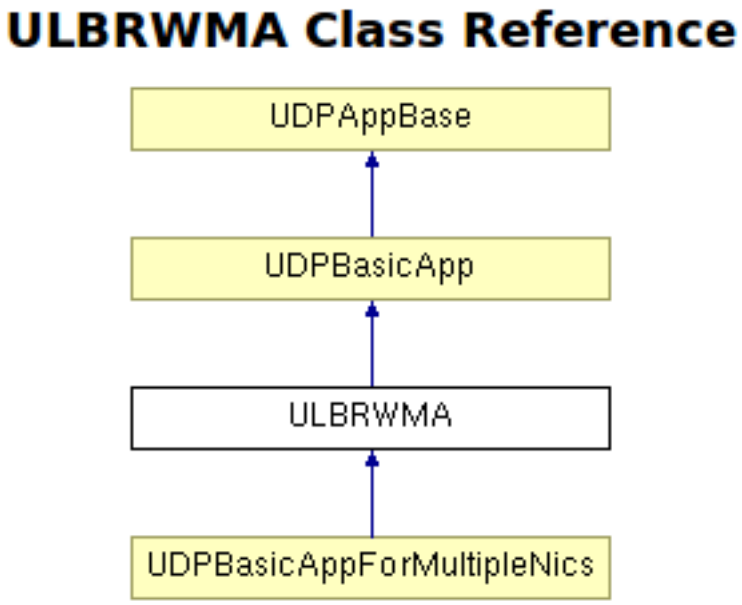

Figura 3.2: ULBRWMA Class Reference

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.

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 da noi utilizzata si differenzia da quest'ultima perché, oltre ad occuparsi di inviare il pacchetto, lo configura, affinché sia riconoscibile dai livelli inferiori come pacchetto RWMA.

[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 sè 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).

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

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 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 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 se, dopo un certo periodo di tempo, non ha 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 dagli 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 un'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 quando 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 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 ai normali pacchetti UDPPacket, IPDatagram e Ieee80211DataFrame utilizzati per il trasporto dei dati, sono presenti due ulteriori tipi di pacchetti per il trasporto delle notifiche all'interno del sistema RWMA.

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

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 la riuscita o il fallimento 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)
- 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 un'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

Obiettivi

Nel nostro lavoro ci siamo concentrati principalmente sul perfezionamento di un modello preesistente, già precedentemente ampliato da Marco Ciamarella [9], dove era stato definito uno scenario simulativo rappresentante una stazione WiFi (host) mobile, che si spostava tra le vie di un ipotetico centro cittadino in presenza di vari Access Point, eseguendo un'applicazione VoIP grazie al meccanismo RWMA. In particolare, vedremo come è stato possibile aumentare il numero degli end-system wireless mobili ad un numero variabile. Inoltre, siamo riusciti a far muovere gli host mobili all'interno del centro cittadino attraverso percorsi casuali e non prestabiliti, come invece accadeva precedentemente.

4.1 Obiettivi della tesi

Gli obiettivi che ci si è prefissati di raggiungere sono:

1. Installazione e configurazione del simulatore di reti open source Omnet++ e il relativo framework Inet su differenti sistemi operativi
2. Modifica del progetto per ottenere un numero di host mobili variabile
3. Modifica del progetto per ottenere percorsi casuali degli host mobili all'interno del centro cittadino

Questo lavoro è stato realizzato insieme alla collega Pamela Rossi.

Capitolo 5

Implementazione

Dopo un'accurata lettura della documentazione a disposizione e un'attenta analisi del progetto da ampliare, è stata stilata una serie di soluzioni. Valutando per ognuna di esse i lati positivi e quelli negativi si è optato per l'implementazione di una soluzione ottimale, che raggiungesse gli obiettivi prefissati e offrisse contemporaneamente una buona efficienza e semplicità nelle revisioni future.

5.1 Installazione

L'installazione e la configurazione del software OmNet++ e del relativo framework Inet sui sistemi operativi WindowsXP e MacOSX è avvenuta correttamente. Sono stati però riscontrati diversi problemi, tra cui la portabilità tra i diversi sistemi e le differenze presenti tra le varie versioni di OmNet++.

5.1.1 Portabilità

È stato inizialmente riscontrato un problema di portabilità tra le varie piattaforme; per risolvere tale problema sono state provate diverse soluzioni, non essendo presente alcuna specifica all'interno della guida di installazione. Infine, si è giunti alla conclusione della necessità di eseguire il comando *make clean* per poter installare correttamente il software, in quanto il compilatore non eseguiva correttamente le operazioni richieste.

5.1.2 Versioni OmNet++

Siamo partiti con l'installazione e l'utilizzo della versione 4.0 di OmNet++ su sistema Windows XP. Non essendo stati riscontrati problemi, si è passati all'installazione su sistema MacOSX, dove si è verificato il problema di portabilità

di cui sopra. Visto che inizialmente non si era riusciti a trovare una soluzione, abbiamo provato a installare le nuove versioni di OmNet++ per vedere se il problema fosse già stato risolto. In realtà, questo ha aggiunto nuove problematiche; infatti, nelle nuove versioni sono state apportate delle modifiche relative alla scrittura del codice come, per esempio, la necessità di assegnare i valori 0 o 1 alle variabili booleane, non riconoscendo più i valori true e false, la gestione dei progetti realizzata in modo differente e altri dettagli implementativi. Infine si è deciso di lavorare sulla versione 4.0 in quanto correttamente funzionante con la precedente implementazione della simulazione, e di cercare di risolvere il problema della portabilità, lasciando come requisito futuro la necessità di apportare le modifiche necessarie per la compatibilità con le nuove versioni del software.

5.2 Programma C++

Per evitare di modificare radicalmente la struttura del precedente progetto, si è optato per la realizzazione di un programma in C++ (denominato randPath.cpp) che eseguito subito prima della simulazione creasse dei file xml contentenenti ogni volta dei nuovi percorsi casuali da associare ai vari host mobili. In questo modo la struttura originaria del progetto viene mantenuta continuando ad utilizzare gli stessi moduli per il movimento (turtleMobile) e la stessa gerarchia.

5.3 Host mobili

Per ottenere un comportamento realistico dello scenario si è riscontrata la necessità di aumentare il numero di host mobili ad un numero variabile e decidibile a tempo di esecuzione dall'utente. Per raggiungere questo obiettivo sono stati modificati i seguenti file (file di configurazione della rete e dei nodi):

- /inet/examples/wireless/voiceoverwifiRWMA/VoWiFiNetworkCity.ned
- /inet/examples/wireless/voiceoverwifiRWMA/rwmaMonitorCity.ini

Con le modifiche apportate il numero di host mobili viene deciso dall'utente al momento dell'esecuzione della simulazione. Per evitare possibili errori da parte dell'utente, è stato limitato a 100 tale valore da dare in input al programma randPath.cpp, che è un numero più che ragionevole per questa simulazione. Il programma modifica il file rwmaMonitorCity.ini inserendo una nuova linea di codice, dopo quelle già presenti, nel modo seguente:

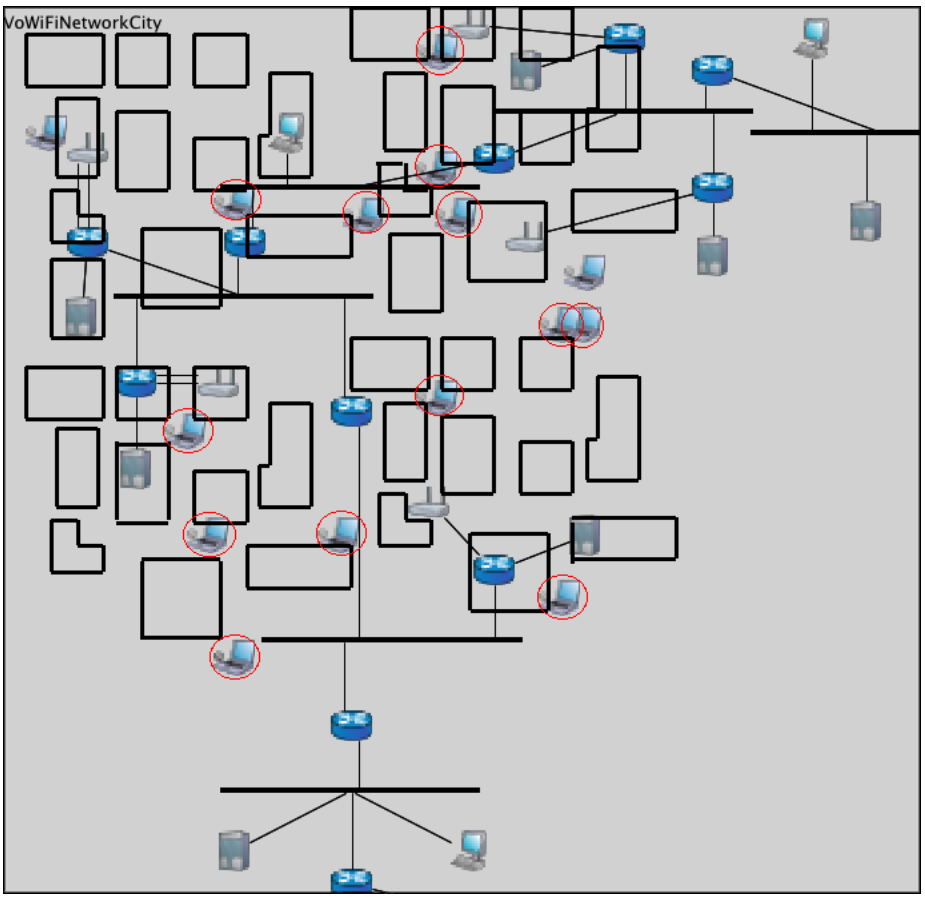

Figura 5.1: Simulazione con 13 host mobili

```
**.mobilehost[K].mobility.turtleScript=
xmldoc("randPath/randPathK.xml","pre//nohtml//movement")
```

per ogni K-esimo mobile host aggiunto.

5.4 Percorsi casuali

Nelle versioni precedenti l'host mobile si muoveva, all'interno del centro cittadino, seguendo un percorso stabilito e fissato a priori. Per un maggior realismo, si è riusciti a far spostare gli host attraverso percorsi casuali. Questo è stato possibile attraverso la creazione dei seguenti file:

- /inet/examples/wireless/voiceoverwifiRWMA/randPath/randPath.cpp
- /inet/examples/wireless/voiceoverwifiRWMA/nodesMap.csv

Il programma contiene le funzioni C++ necessarie per la generazione dei file xml contenenti i percorsi casuali. Per realizzare tali percorsi, sono stati definiti, all'interno del file nodesMap.csv, i punti di incrocio tra le strade del centro cittadino. Questi ultimi sono stati implementati con un'apposita struttura dati contenente quattro elementi, di cui i primi tre sono interi, mentre il quarto è un array di interi. Il primo elemento rappresenta il numero univoco dell'incrocio (id); il secondo e il terzo elemento sono le coordinate x e y del punto; infine è presente l'array contenente i nodi adiacenti al punto stesso.

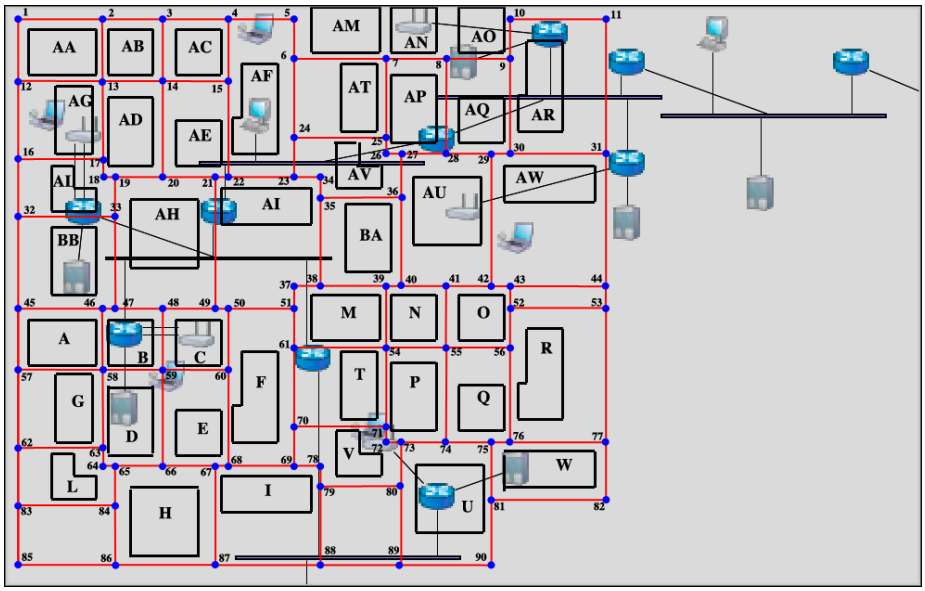

Figura 5.2: Centro Cittadino

Il sistema è stato implementato nel seguente modo: prendiamo ad esemppio l'host mobile K e ipotizziamo che questo si sposti dal punto *1(x1, y1, [2, 12])* al punto *2(x2, y2, [1, 3, 13])*. Una volta raggiunto 2, viene scelto casualmente uno dei punti adiacenti ad esso nel suo vettore delle adiacenze *[1, 3, 13]* dopo aver precedentemente scartato il punto di provenieza 1. La modalità di implementazione descritta è stata scelta affinché il nodo K non possa finire, per pura casualità, in ciclo tra due nodi adiacenti. Ovviamente questa scelta impedirebbe all'host di poter uscire da un vicolo cieco, ma nonostante per nostra scelta non vi siano presenti strade chiuse nel centro abitato, nel caso di modifiche future alla mappa del centro cittadino, non sarà necessario prevedere la gestione di questo caso perché è già stata implementata. L'host mobile K si muoverà quindi da un punto iniziale A scelto casualmente tra 1 e 90, ossia tra tutti gli incroci disponibili, ad un punto finale Z che dista 50 passi da A. Una volta

raggiunto quest'ultimo, K si muoverà a ritroso da Z ad A e ciò verrà ripetuto fino all'interruzione della simulazione.

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 necessario avere sul proprio sistema il framework OMNeT++ 4.0 correttamente compilato e configurato.

6.1 Compilare il codice

Su sistema Windows è possibile usare la shell presente nella cartella di installazione di Omnet++; in questo modo verrà simulato l'ambiente Unix e si potranno usare gli stessi comandi. Oppure, si può utilizzare il classico cmd.exe con i comandi standard dei sistemi Windows.

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.

Invece, se vengono creati dei nuovi file, il makefile va aggiornato lanciando il comando **make -f makemakefiles** e successivamente va invocato 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 con estensione .ned, come nel caso si intenda usare un modulo al posto di un'altro, o file con estensione .ini, non è necessario ricompilare, ma basta lanciare nuovamente la simulazione.

6.2 Simulazioni

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

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

Prima della simulazione, se si vuole generare nuovi perconsi casuali, bisogna spostarsi nella cartella randPath. Successivamente, bisogna compilare il programma randPath.cpp per la propria piattaforma con il comando `g++ randPath .cpp -o randPath`. Infine, si può eseguire tale programma lanciando il comando `./randPath <n>`. Così facendo, verranno creati n file xml contenenti i percorsi casuali. Se si vuole anche visualizzare sul terminale la mappa del centro cittadino e i percorsi generati, bisogna eseguire il comando opzionale `./randPath <n> -print`.

Per lanciare la simulazione, spostarsi nella cartella di installazione di Omnet ed eseguire il programma setenv. In seguito, per eseguire Omnet++, bisogna usare il comando `omnetpp`.

Invece, se si vuole eseguire direttamente la simulazione, bisogna spostarsi nella cartella voiceoverwifiRWMA e lanciare il comando

`./run rwmaMonitorCity.ini`.

È anche possibile generare prima nuovi percorsi e poi lanciare direttamente la simulazione attraverso lo script bash randrun

`./randrun rwmaMonitorCity.ini <n>` dove n è ancora ovviamente il numero di percorsi casuali da generare.

Eseguendo la simulazione coi comandi precendenti, viene mostrata una finestra contenente la configurazione **NetworkCity**:

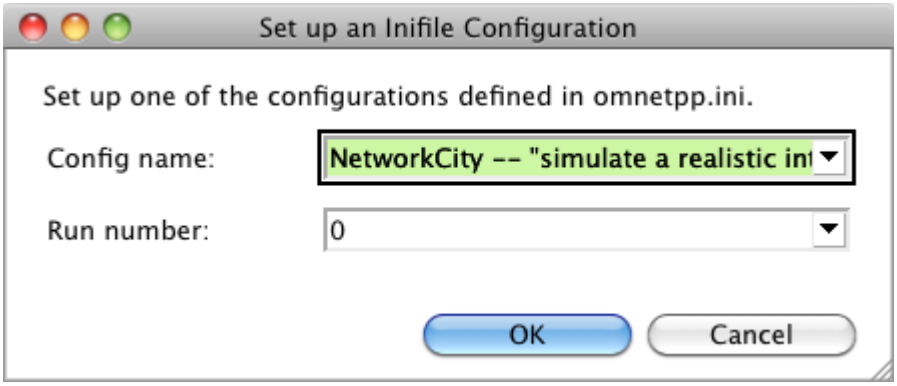

Figura 6.1: Menù di scelta della simulazione

Successivamente, viene mostrata una finestra per la scelta del numero degli host mobili da utilizzare effettivamente nella simulazione:

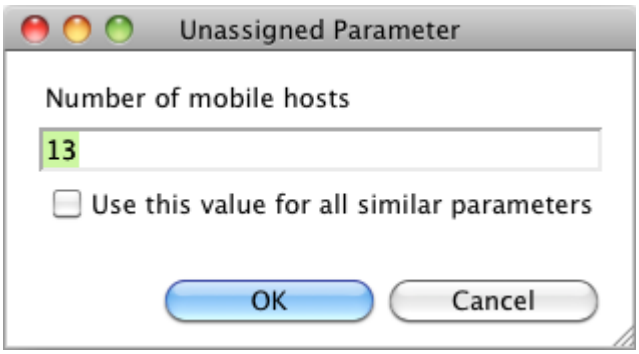

Figura 6.2: Menù di scelta del numero di host mobili

Capitolo 7

Conclusioni

Le modifiche apportate al framework INET e al meccanismo RWMA in esso realizzato, rendono possibile rappresentare con un maggiore grado di realismo il funzionamento di host mobili in contesti di ampia mobilità.

In particolare, è stata resa possibile la realizzazione degli obiettivi prefissati: aumentare il numero dei mobile host da uno ad un numero variabile deciso dall'utente, facendoli spostare all'interno della città tramite percorsi casuali.

7.1 Sviluppi futuri

Per rendere ancora più realistica la simulazione, uno dei possibili miglioramenti da apportare al programma che genera i percorsi casuali è quello di implementare diverse politiche di scelta dei punti iniziali dei mobile host, in modo che vengano determinati non del tutto casualmente ma in modo tale da simulare diverse concentrazioni di utenti in particolari zone del centro cittadino.

In un futuro prossimo grazie a questa simulazione sarà possibile iniziare i test su dispositivi reali e migliorare così la qualità delle trasmissioni VoIP tramite dispositivi mobili.

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/getieee802/download/802.11e-2005.pdf

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

[7] OMNet++ v4.0, http://www.omnetpp.org/omnetpp

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

[9] Marco Ciaramella, “Scenario simulativo per VoIP da dispositivi mobili basato su proxy server SIP/RTP”, Università degli Studi di Bologna, Non pubblicato

[10] 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