ALMA MATER STUDIORUM · UNIVERSITÀ DI BOLOGNA

SCUOLA DI SCIENZE
Corso di Laurea in Informatica

TCP a Ritrasmissione Asimmetrica Anticipata su WiFi

Tesi di Laurea in Reti di Calcolatori

Relatore:
Chiar.mo Prof.
Vittorio Ghini

Presentato da:
Alain Di Chiappari

Sessione III
Anno Accademico 2012/2013

Indice

Elenco delle figure

Capitolo 1

Introduzione

Sentiamo quotidianamente parole come WiFi (IEEE 802.11a/b/g/n), Internet of Things, Access Point Wireless e contemporaneamente ci sentiamo, in modo inevitabile, immersi in un mondo basato sulle informazioni e sulla loro condivisione informatizzata. Abbiamo a portata di mano in ogni momento della giornata dispositivi elettronici sempre più *general purpouse*[1] dotati di connettività wireless che ci permettono di trasmettere e ricevere dati per gli scopi più disparati, dalla semplice applicazione ludica sullo smartphone, fino a dispositivi medici [3] per il monitoraggio e la rilevazione d'urgenza di parametri clinici del paziente.
Uno dei principali nemici delle tecnologie WiFi (oltre alla sicurezza) sono le interferenze ambientali; l'esperirenza stessa e la ricerca in questo settore ci dicono che sono numerosi i fattori d'ostacolo alle prestazioni della connettività wireless, partendo dai muri portanti in cemento armato ai metalli in genere, per arrivare a vere e proprie fonti di interferenza sullo spettro elettromagnetico, quindi dai diffusissimi forni a microonde, telefoni cordless, ripetitori video e alcuni monitor, passando per altoparlanti, mouse, tastiere e cuffie wireless qualora questi siano nelle prossimità di una scheda di rete, fino ad arrivare a fonti di disturbo esterne come sistemi antifurto, antenne paraboliche, cavi dell'alta tensione, centrali elettriche e così via. Vi è infine l'interferenza fra reti wireless impostate su canali uguali o vicini, come sarà accennato in seguito.
Tutto ciò influenza in misura più o meno grave la qualità della connettività wireless, quindi la portata del segnale, la perdita di connessione tra stazioni e access point, la diminuzione complessiva del throughput[2] con la conseguente percezione di "rallentamento" della connessione e così via.

[1] Sono general purpouse quei device elettronici come il classico Personal Computer che non fornisce un numero ben definito di funzionalità come può essere un lettore mp3, uno stereo, un televisore. I dispositivi general purpouse solitamente danno la possibilità di caricarvi del software per svolgere svariate funzioni; negli ultimi anni la linea di demarcazione tra dispositivi general purpouse e non, è diventata molto sottile, basti pensare a cosa possiamo fare con un "cellulare", la possibilità di usare il televisore per gestire la domotica di un'abitazione e così via.

[2] Si intende per throughput la quantità di dati effettivamente trasmessi in una certa unità di tempo, è quindi usato come uno dei parametri della qualità di un collegamento di rete in genere.

In quest'ottica nasce e viene inserito nel kernel Linux, **RWMA** (il meccanismo **Robust Wireless Medium Access**), con lo scopo di aumentare la **Quality of Service** di un nodo di rete mobile (come può essere uno smartphone) immerso nello scenario urbano.
RWMA (e precedentemente, a cui è collegato, ABPS[3]) è stato implementato per aumentare la QoS di sistemi VoIP su WiFi, quindi basata sul protocollo di trasporto UDP; è stato inoltre sviluppato un meccanismo di QoS sul protocollo TCP (su WiFi), andando ad agire sulla ritrasmissione dal nodo mobile all'Access Point di quei dati persi nella sottorete wireless. Ciò che è stato fatto in questo elaborato è il completamento speculare del lavoro sul TCP, si è quindi agito sull'Access Point per "recuperare" i dati persi nell'ultimo hop di trasmissione fino al nodo mobile, facendo poi un'analisi per via simulativa del meccanismo implementato.
Come appenna accennato, data la difficoltà, in termini di costo, di tempo e di tecnologie, si è sviluppato ed analizzato il meccanismo di ritrasmissione anticipata su Access Point nell'ambiente simulativo OMNET++ congiuntamente al framework INET; **OMNET++** è un ambiente di sviluppo modulare ad eventi e scambio di messaggi, che ci consente di creare simulazioni in qualsiasi ambito; **INET** è invece di fondamentale importanza in quanto fornisce una serie di protocolli, schede e nodi di rete per lo sviluppo di simulazioni nel mondo del networking.

L'elaborato è stutturato come segue:

- Nei primi due capitoli vengono esposti i due concetti su cui si basa il meccanismo sviluppato, il TCP, protocollo di trasporto sul quale si vuole ottenere un beneficio in termini di miglioramento della QoS, ed in secondo luogo il protocollo IEEE802.11, sul quale si andrà effettivamente ad agire per lo scopo perseguito.

- Si passerà poi, a descrivere il TCP a ritrasmissione anticipata sul nodo mobile, al fine di definirne i concetti e introdurre il meccanismo qui sviluppato, sull'Access Point.

- Nel capitolo successivo si descrivono gli obiettivi dell'elaborato, alla luce del contesto nel quale è sviluppato, quindi si definiranno gli scopi e l'ambito di applicazione, oltre ai motivi e le problematiche da cui nasce RWMA e la ritrasmissione anticipata.

- In seguito ci si focalizzerà sugli aspetti progettuali, quindi le scelte implementative e la loro applicazione pratica nel framework di sviluppo simulativo OMNET++/INET. Nello

[3]L'architettura ABPS (Always Best Packet Switching) [6] viene introdotta per poter sfruttare al massimo le potenzialità di un dispositivo in modo da utilizzare nello stesso tempo tutte le interfacce di rete di cui è dotato il nodo mobile, instradando il pacchetto da inviare su quella che, al momento dell'invio, si reputa essere l'interfaccia di rete migliore; in questo modo, ABPS permette alle applicazioni mobili di creare politiche di bilanciamento di carico

specifico si descriveranno le modifiche ai vari moduli, oltre a presentare una possibile alternativa progettuale di sviluppo.

- Vi sarà inoltre un capitolo riguardante gli aspetti più tecnici, quindi dall'elenco dei file modificati nel progetto, fino ai comandi necessari per la fase di compilazione ed esecuzione del codice.

- Il capitolo sulle valutazioni mostrerà, a partire dai criteri con il quale si farà un'analisi, e in base allo scenario della sperimentazione, i risultati e le misurazioni di alcune prove simulative. Verrano presentati questi test, focalizzandosi su alcuni parametri variabili, grazie ai quali si procederà ad un'analisi di efficienza ed efficacia della ritrasmissione anticipata.

- Nell'ultimo capitolo si raccoglieranno alcune riflessioni su quanto esposto nei capitoli precedenti, comprese le problematiche sollevate dallo sviluppo del meccanismo di QoS. Infine si introdurranno alcuni possibili sviluppi futuri, ottimizzazioni o migliorie, che si ritengono utili alla prosecuzione dei fini illustrati.

Capitolo 2

Trasmission Control Protocol

2.1 Caratteristiche generali

Il TCP [2,5,6] (Transmission Control Protocol) è un protocollo di livello trasporto. A differenza di UDP, è orientato alla connessione (o connection oriented), poiché due nodi, prima di trasmettere dati tra loro, devono instaurare una connessione; la fase di inizio della connessione, in cui vengono trasferite informazioni preliminari, per poi poter garantire un servizio affidabile di consegna dei pacchetti, viene detta three-way-handshake (o stretta di mano in tre passi).
TCP è un protocollo stream oriented, ovvero i dati trasmessi vengono visti come un flusso di byte ordinati e numerati, inoltre, offre un servizio full-duplex, ovvero il mittente può inviare dati al destinatario e viceversa, ma più precisamente i flussi di dati con direzioni diverse possono coesistere contemporaneamente; ciò significa che un nodo A può inviare pacchetti a un nodo B e nello stesso tempo può ricevere pacchetti da un nodo C. Ciò che non è possibile fare, invece, è un invio multicast: un mittente non può trasferire dati a più destinatari in una sola operazione; se ipotizziamo che un generico host debba notificare qualcosa ad altri tre host, esso dovrà necessariamente instaurare una connessione TCP separata per ogni destinatario. La connessione è point-to-point, cioè relativa ad una sola sorgente e una sola destinazione. Le caratteristiche principali del TCP sono:

- trasferimento affidabile dei dati: tutti i dati inviati da un mittente sono recapitati al destinatario senza errori. Può succedere che, per vari motivi (errore tra i collegamenti, disturbi esterni alla comunicazione, etc.), qualche pacchetto può contenere errori oppure vada perduto nella rete. A questo punto il TCP si occuperà di rilevare questa mancata ricezione e procederà a ritrasmettere il pacchetto perso; questa caratteristica è presente in TCP e non in UDP che, invece, non garantisce un trasferimento affidabile dei dati;

- corretto ordinamento dei pacchetti: può accadere che i pacchetti contenenti le informazioni subiscano ritardi all'interno della rete (per via del congestionamento della stessa o

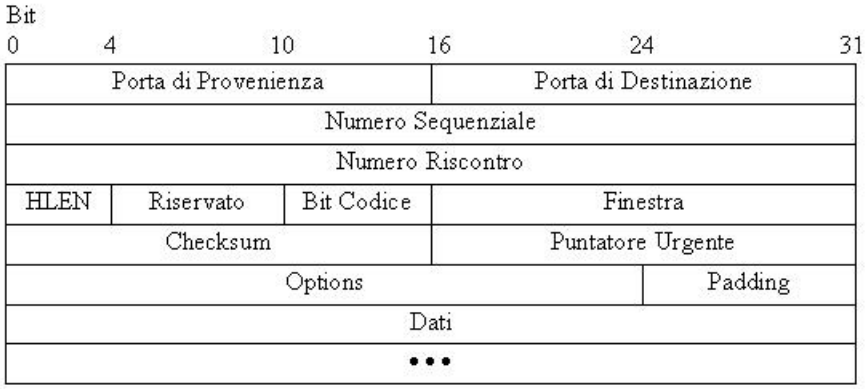

Figura 2.1: Frame TCP

perché due pacchetti possono seguire due rotte differenti per arrivare a destinazione, etc.) con il conseguente risultato che arrivano a destinazione non in ordine. Il compito del TCP è proprio quello di riassemblare il messaggio in maniera corretta, ordinando i pacchetti per il numero di sequenza contenuto nei pacchetti stessi e inoltrarlo al livello superiore;

- controllo del flusso: il TCP stabilisce se un host più veloce nella trasmissioni dati rischia di mandare in overflow il buffer di ricezione di un host più lento. Può accadere, infatti, che l'applicazione mittente generi i pacchetti con una frequenza molto maggiore rispetto a quella con cui l'host destinazione riesce a ricevere, con il conseguente riempimento del buffer di ricezione e della successiva eliminazione dei pacchetti da parte del destinatario. Per evitare questo, il TCP abbassa la frequenza di trasmissione del mittente al fine di equilibrare le prestazioni dei due host;

- controllo della congestione [4,7]: anche in questo caso, il TCP si accorge che c'è un congestionamento diffuso della rete, e quindi impone all'host mittente di diminuire la frequenza di trasmissione dei pacchetti per evitare di aggravare la situazione.

2.2 Il Pacchetto TCP

La struttura di un pacchetto TCP prevede:

- i campi Source Port e Destination Port, necessari per il corretto multiplexing e demultiplexing da parte delle singole applicazioni su un host;

- il campo Checksum viene utilizzato per capire se un pacchetto è arrivato a destinazione con errori di trasmissione oppure no;

- i campi Sequence Number e Acknowledgement Number, entrambi di lunghezza pari a 32 bit, contengono rispettivamente il numero di sequenza del pacchetto che si sta inviando e

il numero del pacchetto che è stato ricevuto e di cui si sta comunicando l'acknowledgement. Sono i campi più importanti dell'header di un pacchetto TCP in quanto su di essi si basa il trasferimento affidabile dei dati; si ipotizzi, ad esempio, di dover trasmettere dal nodo A al nodo B un messaggio di lunghezza 2000 byte e che il MSS (Maximum Segment Size) sia fissato a 500 byte. Il messaggio viene frammentato in quattro segmenti; il numero di sequenza del primo segmento sarà 0, del secondo segmento 500, del terzo segmento 1000 e infine del quarto 1500. Il campo Aknowledgement Number viene utilizzato dal nodo B verso il nodo A in risposta all'arrivo dei frammenti inviati; ad esempio, il nodo A invia il primo segmento (di 500 byte) al nodo B (frammento con sequence number posto a 0), il quale riceve il frammento correttamente e invia un pacchetto di ACK al nodo A, contenente nel campo Acknowledgement Number il valore 501 e così via per i frammenti successivi.

Si consideri, ora, un'altra situazione, ovvero lo scenario in cui l'host di destinazione abbia ricevuto correttamente il primo (0-500 byte) e il terzo (1000-1500 byte) frammento, ma non il secondo (500-1000 byte). In questa situazione anomala l'host destinatario scriverà nel campo Acknowledgement Number del pacchetto di ACK il numero 501, in quanto il campo fa riferimento al primo byte successivo tra quelli arrivati in maniera ordinata. Una volta ricevuto il secondo pacchetto (500-1000 bytes), il destinatario, accorgendosi che il terzo frammento è già stato inviato, scriverà nel numero di riscontro il valore 1501 che fa riferimento al quarto e ultimo segmento dei dati totali.

- il campo Header length è formato da 4 bit ed è usato per specificare la lunghezza totale dell'intestazione del pacchetto; viene utilizzato quando il campo opzioni ha dimensioni variabili, e quindi permette di calcolare la grandezza dell'intestazione per ogni singolo pacchetto;

- il campo Window Size è un campo di lunghezza 16 bit ed è viene utilizzato per il controllo di flusso descritto in seguito;

- il campo Option è l'unico facoltativo, ha una lunghezza variabile e viene utilizzato per l'abilitazione di particolari versioni del protocollo;

- il campo Flag ha lunghezza 8 bit ed ognuno di essi può essere visto come un campo a sé stante. Proprio questi flag vengono utilizzati per implementare il servizio di ritrasmissione in caso di perdita e per effettuare l'handshake iniziale. Il bit ACK indica, se settato a 1, che il segmento ingloba un riscontro di un pacchetto ricevuto precedentemente senza errore. SYN, FIN, RST (assieme ad ACK) vengono utilizzati per il processo di handshake e per la chiusura della connessione. Il campo PSH, poco usato, impone all'host che riceve il

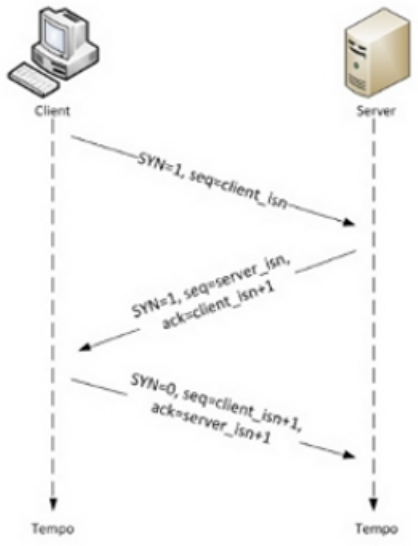

Figura 2.2: Handshake a tre vie

pacchetto di inoltrarlo immediatamente al livello applicazione. Il campo URG, anch'esso poco usato, indica all'host di destinazione che il segmento contiene dei pacchetti che il livello applicativo mittente ha marcato come urgenti.

2.3 Handshake a tre vie

Il TCP è un protocollo connection oriented, ovvero due nodi che vogliono comunicare tra loro, prima di poter trasferire dei dati devono instaurare la connessione. Questa viene stabilita attraverso un algoritmo chiamato handshake a tre vie, in quanto consta di tre fasi principali:

- il client invia al server un segmento vuoto, senza dati, con il bit SYN del campo Flag posto a 1 e con valore del campo sequence number inizializzato a un valore CLIENT_ISN, scelto in maniera pseudocasuale tra un range di valori possibili;
- il server riceve il segmento e risponde al client con un altro segmento vuoto con i bit SYN e ACK posti a 1, con campo di riscontro (Acknowledgement Number) settato al valore CLIENT_ISN+1 e con numero di sequenza settato a un altro valore iniziale SERVER_ISN, scelto questa volta in modo pseudocasuale dal server.
- il client invia al server un nuovo segmento vuoto, senza dati, con il bit SYN posto a 0 e ACK posto a 1, con campo di riscontro (Acknowledgement Number) settato al valore SERVER_ISN+1 e con numero di sequenza settato al valore CLIENT_ISN+1.

Lo stesso algoritmo può essere utilizzato (sostituendo al bit SYN, il bit FIN) anche per la chiusura della connessione, qualora si opti per una chiusura *bidirezionale*; in TCP vi è però

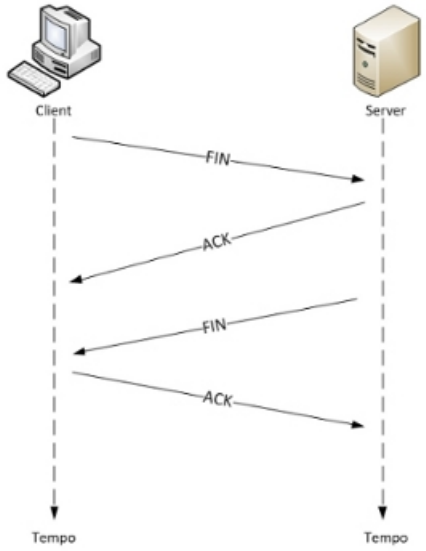

Figura 2.3: Handshake a quattro vie (FIN+ACK)

la possibilità di chiudere solo un lato della connessione, mentre l'altro endpoint può e deve continuare a ricevere dati, fin quando non chiude anch'esso dal suo lato. In questo caso, la chiusura non è bidirezionale e quindi consta di quattro fasi ed è per questo che viene chiamato handshake a quattro vie.
In particolare:

- il client invia al server un segmento con il bit FIN del campo Flag settato a 1, e si pone nello stato FIN_WAIT_1;
- il server invia al client un riscontro e il client, una volta ricevuto ACK, entra nello stato FIN_WAIT_2;
- il server invia al client un segmento con il bit FIN del campo Flag settato a 1;
- il client riscontra il segmento inviando un ACK al server ed entra nello stato TIME_WAIT; dopo un tempo variabile la connessione viene chiusa.

2.4 Controllo di flusso

Si supponga una comunicazione tra un nodo A e un nodo B; nel momento in cui il nodo A invia un pacchetto al nodo B questo, al momento della ricezione, viene posto in un buffer di ricezione, inizializzato durante la fase di handshake e letto periodicamente dall'applicazione, che ne processa il contenuto.
Se la lettura dell'applicazione sul nodo B è molto più lenta dell'invio da parte del nodo A si può verificare un overflow del buffer di ricezione; il controllo di flusso serve proprio ad

evitare questo problema. Entrambi i nodi tengono traccia all'interno di una variabile, detta finestra di ricezione (o Received Windows) dello spazio libero del buffer di ricezione nel nodo corrispondente, secondo la quale creano ed inviano una quantità di dati che evita l'overflow del buffer. Qualora si ottenga un pacchetto in cui il campo della finestra di ricezione ha valore pari a 0, si interrompe la trasmissione, tranne per un pacchetto da 1 byte in modo tale da permettere alla destinazione di aggiornare il mittente con il pacchetto ACK riguardo la sua situazione.

2.5 Trasferimento affidabile dei dati

Il TCP garantisce che tutti i dati inviati da un mittente sono recapitati al destinatario senza errori; lavorando in una rete, per di più wireless, si possono verificare eventi che non garantiscono il trasferimento affidabile dei dati.
Tra questi, i principali sono:

- corruzione: il pacchetto ricevuto dal destinatario contiene degli errori;
- timeout: il segmento inviato dal mittente viene ricevuto dal destinatario "fuori orario" oppure non viene ricevuto per niente;
- arrivo fuori ordine: i pacchetti, inviati dal mittente verso il destinatario, possono seguire rotte differenti nella rete e quindi potrebbero arrivare in un ordine diverso rispetto a quello utilizzato dal mittente in fase di trasmissione.

Il TCP attua diversi accorgimenti per ovviare a questi errori:

1. riscontro: nel momento in cui il destinatario riceve un pacchetto corretto, il ricevente notifica al mittente l'avvenuta ricezione attraverso uno speciale pacchetto detto ACK (Acknowledgement); la mancata ricezione di un segmento ACK può causare l'invio di pacchetti duplicati;
2. numeri di sequenza: sia i segmenti che gli ACK hanno dei numeri identificativi univoci all'interno del campo sequence number; questo permette di comunicare quale o quali segmenti contengono errori, quali sono arrivati fuori ordine o non sono arrivati affatto, etc.;
3. timer: segmenti e ACK possono essere persi durante la trasmissione. TCP gestisce questa fonte di errori impostando un timeout quando vengono inviati i dati; nel caso in cui il timeout scada prima dell'arrivo del riscontro, i segmenti vengono inviati nuovamente. TCP gestisce quattro diversi timer per ogni connessione.

- retransmission timer: gestisce i timeout di ritrasmissione, che si hanno quando si è terminato l'intervallo tra l'invio di un datagramma e la ricezione del segmento acknowledgement. Il valore del timeout non è statico, ma viene calcolato dinamicamente a seconda della velocità e della congestione della rete; in particolare viene determinato misurando il tempo necessario al segmento per arrivare al destinatario, aggiungendo quello dell'ACK per ritornare al mittente; questo tempo viene chiamato round trip time (RTT). Nel protocollo viene effettuata una media dei RTT e ne consegue un valore atteso chiamato smoothed round-trip time o SRTT.

  Nel momento in cui viene inviato un datagram, il retransmission timer viene settato con il valore SRTT calcolato; se l'ACK arriva entro lo scadere del timer, ciò presuppone che il pacchetto è arrivato a destinazione, allora il timer viene annullato; mentre se il timer scade, il datagram viene inviato nuovamente e il timer settato ancora con un RTO (Retransmission Timeout) modificato, che usualmente si incrementa esponenzialmente fino a un limite massimo, oltre il quale si presume un fallimento della connessione.

- timed wait: nel momento in cui viene interrotta la connessione TCP, è possibile che alcuni datagram tentino di accedere alla porta legata alla connessione terminata; il timed wait ha lo scopo di impedire che la porta appena chiusa venga aperta di nuovo per ricevere questi datagrammi, che, conseguentemente, vengono scartati. Questo timer è usualmente impostato al doppio del tempo di vita massimo di un segmento (lo stesso valore del campo Time to live in un header IP).

- persist timer: gestisce un evento molto raro. Come precedentemente detto, si può verificare la situazione in cui il buffer di ricezione del nodo destinatario sia colmo e quindi che il campo windows size contenga il valore zero, costringendo il nodo mittente a mettere in pausa la trasmissione. Si è, inoltre, specificato che per riattivare la comunicazione, il nodo mittente manda un messaggio di un byte a cui il nodo destinatario risponde inviando la dimensione del buffer, oramai svuotata; tuttavia, si può verificare che questi messaggi vengano persi, causando un ritardo infinito nel riavvio della comunicazione.

  Il persist timer viene utilizzato per fare in modo che nel caso in cui il messaggio di un byte venga perduto, il nodo attenda lo scadere del timer e poi invii nuovamente il segmento di un byte ad intervalli predeterminati, per assicurarsi che la macchina riceva il messaggio e per stabilire se è ancora bloccata. La macchina ricevente rinvia il messaggio di notifica per la finestra di ampiezza zero ogni volta che riceve uno di

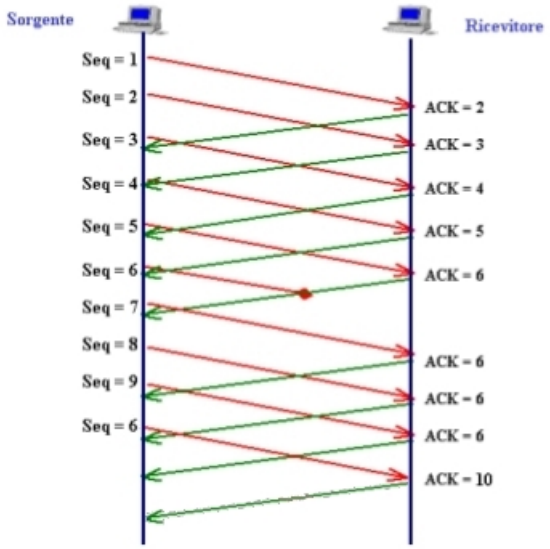

Figura 2.4: Trasferimento dati e controllo di errore in TCP

questi segmenti di stato se è ancora intasata, altrimenti inoltra un messaggio con il nuovo valore di finestra e la comunicazione viene ripristinata.

- keep-alive timer: manda un pacchetto vuoto a intervalli regolari per assicurare che la connessione con l'altra macchina sia ancora attiva. Se non si è ricevuta risposta dopo che è stato inviato il messaggio, prima che scada il timer di idle, si presume che la connessione sia interrotta. Il keep-alive timer è usualmente impostato da un'applicazione con valori che variano fra 5 e 45 secondi. Il timer di idle è usualmente impostato a 360 secondi.

2.6 Controllo della congestione

Una delle principali cause di degradazione delle prestazioni delle reti è senza dubbio la congestione, che si verifica quando per un certo intervallo di tempo il traffico offerto generato dai terminali è maggiore della possibilità di smaltirlo da parte della rete stessa. Infatti, ogni volta che un nodo interno alla rete riceve dati più velocemente di quanto riesca a smaltirli, è costretto a creare una coda e mettere alcuni pacchetti in attesa. Pur senza entrare nello specifico della gestione delle code e della teoria del traffico, è facile intuire quanti inconvenienti possano creare alle prestazioni della rete il perdurare di una situazione di congestione in diversi nodi. Oltre ad aumentare in modo notevole il ritardo dei pacchetti, può infatti succedere che dei buffer si saturino e scartino dei pacchetti: il terminale sorgente provvederà dunque a ritrasmetterli, aumentando ancora di più il carico offerto alla rete. Ancora peggiore è il caso in cui un pacchetto

che passa molto tempo nella coda di un router venga ritenuto perso dall'host che lo ha generato (a causa dello scadere del Retransmission Timeout): il mittente lo ritrasmetterà anche se in realtà non è necessario, sprecando così notevoli risorse.

Facciamo qui una piccola digressione storica sulle versioni del TCP riguardo al problema della congestione, essendo questo un punto importante per il meccanismo precedentemente sviluppato di ritrasmissione anticipata sul nodo mobile.
Le diverse versioni di TCP che sono state sviluppate nel corso degli anni si differenziano principalmente proprio per il modo di reagire a degradazioni delle prestazioni della rete, siano esse dovute a congestioni o a rumorosità del canale. Nelle prime versioni del TCP (fino al 1986), sostanzialmente non veniva preso alcun provvedimento per evitare i problemi causati dalle congestioni. Nel 1986 fu introdotto il TCP Berkeley, che cominciava ad affrontare il problema introducendo gli algoritmi Slow Start e Congestion Avoidance [8]. Successivamente, nel 1988, il TCP Tahoe aggiunse a questi anche l'algoritmo Fast Retransmit [8], e nel 1990 con il TCP Reno fu introdotto l'algoritmo Fast Recovery [8]. Nel 1995 questo venne leggermente modificato (TCP New Reno [9]), e per molti anni rimase il protocollo più utilizzato nella rete. Il TCP Vegas [11] introdurrà una nuova filosofia, cercando di prevenire la congestione e rallentando il ritmo di invio dei dati prima che si verifichino eventi di perdita, ma questo lo renderà poco compatibile con le altre versioni.
Con l'avvento delle wireless LAN furono sviluppate nuove versioni che meglio si adattavano al nuovo canale, come TCP Westwood [10] e Veno [14]. L'approccio seguito dal TCP per effettuare il controllo della congestione è di fare si che ogni mittente limiti il ritmo con cui immette traffico nella sua connessione in funzione della congestione in rete percepita. Ovviamente, se un terminale percepisce che c'è poca congestione nel percorso tra sé e la destinazione, il TCP aumenta il suo ritmo di trasmissione, mentre lo riduce se sente che alcuni nodi sono congestionati.
Il primo argomento da affrontare è sapere come il TCP si accorge che la rete è congestionata e come fa per regolare il ritmo di trasmissione. Si consideri in prima battuta come il mittente regola la quantità di dati che immette nella rete: per fare ciò il TCP utilizza una nuova variabile detta finestra di congestione (Congestion Window). La finestra di congestione (Cwnd) impone una limitazione addizionale alla quantità di traffico che un host può inviare, oltre a quella che già abbiamo analizzato per il controllo del flusso. Riprendendo la terminologia precedentemente utilizzata, l'ammontare di dati non riscontrati che un terminale può avere durante una connessione TCP risulta:

$$LastByteSent - LastByteAcked \leq minCwnd, RcvWin$$

Per capire se la rete è congestionata o meno, il TCP utilizza i cosiddetti eventi di perdita, che possono essere o lo scadere di un timeout o, nelle versioni con l'algoritmo Fast Retransmit, la ricezione di tre ACK duplicati: quando c'è congestione, infatti, uno o più buffer dei router lungo il percorso si possono saturare fino a dover scartare dei pacchetti. Il mittente si accorge del datagram perso e quindi della congestione grazie appunto agli eventi di perdita e può dunque agire di conseguenza. Come già accennato, il modo di reagire a queste situazioni cambia a seconda delle versioni del TCP.

2.7 TCP in INET/OMNET++

Il modulo INET del TCP [6] implementa questo protocollo di trasporto nella sua totalità; è connesso con il livello IP con due tipologie di gates:

- ipIn e ipOut, che rappresentano rispettivamente il gate di ingresso e di uscita per il livello IPv4.
- ipv6In e ipv6Out, che rappresentano rispettivamente il gate di ingresso e di uscita per il livello IPv6.

Inoltre, è connesso al livello applicazione attraverso l'array di gates appIn[] e appOut[]; il modulo TCP può servire svariate applicazioni contemporaneamente: il modulo dell'applicazione k-esima è connesso al modulo TCP attraverso le porte appIn[k] e appOut[k]. Applicazione e modulo TCP comunicano tra loro attraverso lo scambio di messaggi; l'applicazione invia al modulo TCP messaggi di tipo TCPCommand, come TCP_C_OPEN_ACTIVE, TCP_C_OPEN_PASSIVE, TCP_C_SEND, TCP_C_CLOSE, TCP_C_ABORT, TCP_C_STATUS.
Ognuno di questi messaggi, che rappresentano essenzialmente dei comandi che l'applicazione impone al livello TCP, contiene al suo interno delle informazioni di controllo ed un identificatore univoco, i quali individuano la connessione con l'applicazione. Il modulo TCP risponde ai comandi inviati dall'applicazione attraverso messaggi di tipo TCPStatus.
Anche questi messaggi, che rappresentano essenzialmente lo stato in cui si trovano il modulo e la connessione TCP, vengono inviati all'applicazione insieme ad alcune informazioni di controllo.
A questo punto, è evidente come il TCP sia gestito e implementato attraverso una macchina a stati finiti, in cui la classe TCPConnection gestisce e implementa essa stessa la macchina a stati, mentre la classe TCPAlgorithm implementa i vari algoritmi di controllo della congestione visti in precedenza. In particolare, la classe TCPConnection gestisce le transizioni da uno stato ad un altro, date dal verificarsi di un evento, e memorizza lo stato della macchina (TCB) contenente informazioni essenziali quali:

- il campo active per indicare se la connessione è attiva o passiva;
- il campo fork utilizzato quando la connessione è passiva, per indicare se è stata fatta una fork della connessione;
- il campo snd_mss rappresentante il maximum segment size;
- i campi snd_una, snd_nxt, snd_max rappresentanti rispettivamente il numero di sequenza del primo pacchetto di cui non è stato ricevuto ancora ack, il numero di sequenza del prossimo pacchetto da inviare e il massimo numero di sequenza dei pacchetti inviati (questo perchè snd_nxt può essere portato indietro in caso di ritrasmissione);
- il campo snd_wnd per indicare la finestra di invio;
- il campo snd_up per indicare il puntatore al pacchetto urgente;
- il campo snd_iss contenente il valore per l'initial sequence number;
- i campi snd_wl1 e snd_wl2 contenenti rispettivamente il numero di sequenza usato per inviare l'ultima finestra aggiornata e il numero di ack usato per inviare l'ultima finestra aggiornata;
- i campi rcv_nxt, rcv_max e rcv_up rappresentanti rispettivamente il numero di sequenza del prossimo pacchetto da ricevere, il valore della finestra di ricezione e il numero di sequenza di un pacchetto urgente ricevuto;
- il campo irs contenente il valore per l'initial receive sequence number;
- il campo dupack contentente il numero di ack duplicati ricevuti per un pacchetto;
- i campi syn_rexmit_count, sys_rexmit_timeout, fin_ack_rcvd, send_fin, snd_fin_seq, fin_rcvd e rcv_fin_seq utilizzati per gestire le fasi di handshake durante l'instaurazione della connessione.

Questi campi permettono di implementare una versione base del protocollo TCP, a cui ne vanno aggiunti eventualmente degli altri nel caso in cui i vari algoritmi di controllo della congestione ne abbiano bisogno. Infine, ci sono ancora due "oggetti" che entrano in gioco e vengono gestiti anch'essi da TCPConnection: questi sono le istanze di TCPSendQueue e TCPReceiveQueue, rappresentanti rispettivamente i buffer di invio e di ricezione dei messaggi.
Infine, si introduce la classe TCPSocket, più di livello applicazione che di livello trasporto (e quindi TCP), ma che permette una gestione semplice della connessione TCP; in particolare si hanno uno o più oggetti rappresentante il socket TCP nell'applicazione e possono essere usati i suoi metodi per aprire o chiudere una connessione TCP, o per inviare e ricevere messaggi

da/verso il modulo TCP.
Le operazioni principali svolte nel modulo TCP sono:

2.7.1 Instaurazione della connessione

L'instaurazione della connessione può essere fatta in due maniere differenti:

- l'applicazione apre una porta locale per una connessione in ingresso inviando un messaggio TCP_C_PASSIVE_OPEN al modulo TCP, contenente l'indirizzo e la porta locali. L'applicazione può, inoltre, specificare se gestire una sola connessione TCP alla volta o più connessioni multiple: se il campo fork è settato a true, allora viene creato un thread con un nuovo id e, mentre il thread figlio gestisce la connessione che si sta creando, il thread padre si pone in uno stato di listening per accettare altre eventuali connessioni; se il campo fork è settato a false, allora viene gestita una sola connessione alla volta.

- l'applicazione apre una connessione inviando un messaggio TCP_C_ACTIVE_OPEN al modulo TCP; questo crea un oggetto TCPConnection e invia un segmento SYN; a questo punto ha inizio la fase di handshake a 3 vie. Dopo 75s, se la fase di instaurazione della connessione non è terminata, scatta il timeout, viene inviato TCP_I_TIMEOUT all'applicazione e viene chiusa la connessione. Se, invece, la connessione viene instaurata, TCP invia all'applicazione TCP_I_ESTABLISHED, insieme all'indirizzo e porta remoti; altrimenti, si può verificare il caso in cui la connessione venga rifiutata dall'host remoto, per cui il TCP invia TCP_I_CONNECTION_REFUSED all'applicazione.

2.7.2 Invio dei pacchetti

L'applicazione invia i pacchetti al modulo TCP, insieme al comando TCP_C_SEND; il modulo TCP aggiunge il pacchetto nella coda di invio, controlla se il pacchetto può essere spedito oppure no, lo divide in frammenti (nel caso in cui la dimensione totale del pacchetto sia maggiore del maximum segment size) e invia questi ultimi, sottoforma di TCPSegment, al modulo IP. Esistono tre tipologie differenti di code di invio, a seconda delle tre modalità di trasferimento dei dati:

- TCP_TRANSFER_BYTECOUNT: il segmento TCP ha payload nullo;

- TCP_TRANSFER_BYTESTREAM: l'applicazione invia al modulo TCP un array di byte; il modulo divide l'array in pezzi di dimensioni pari alla MSS e li trasmette nel campo payload del segmento TCP;

- TCP_TRANSFER_OBJECT: l'applicazione passa al modulo TCP un oggetto di tipo cMessage; il TCP crea i segmenti necessari, in base alla lunghezza massima consentita, e invia l'oggetto come payload del primo segmento.

Per quanto riguarda le operazioni di ritrasmissione del pacchetto, di controllo di flusso e di congestione, queste vengono gestite dalla classe TCPAlgorithm, in base all'algoritmo scelto.

2.7.3 Ricezione dei pacchetti

La connessione TCP memorizza i segmenti in arrivo nella coda di ricezione; come per il lato trasmittente, anche per il lato ricevente esistono tre tipologie di code differenti, in base al tipo di trasferimento dei dati fatto dal trasmittente. Durante la fase di ricezione, se il buffer è pieno o la dimensione del pacchetto è maggiore dello spazio libero sul buffer, il messaggio viene scartato; inoltre, se il pacchetto ricevuto ha il sequence number successivo all'ultimo ricevuto (e quindi è il pacchetto che si stava aspettando), questo viene subito inoltrato all'applicazione incapsulato nel messaggio TCP_I_DATA, altrimenti viene bufferizzato in attesa di essere ricomposto e solo successivamente passato all'applicazione.

2.7.4 Chiusura della connessione

Come per l'instaurazione, anche la chiusura della connessione può essere fatta in due modalità differenti.

- l'applicazione lato client decide di chiudere la connessione, in quanto non ha più dati da inviare; comanda, attraverso il TCP_C_CLOSE, al modulo TCP di cessare la connessione. Il modulo TCP tenta di inviare tutti i pacchetti all'interno del suo buffer di trasmissione e solo successivamente inizia la fase di chiusura della connessione, ovvero le operazioni viste nell'algoritmo di handshake a 3 o 4 vie (FIN+ACK);
- il server decide di chiudere la connessione; il modulo TCP lato client riceve un segmento FIN e invia un messaggio TCP_I_PEER_CLOSED alla applicazione client, contenente l'identificatore della connessione e altre informazioni di controllo.

Durante la comunicazione, tuttavia, si possono verificare degli eventi per cui la connessione può essere resettata oppure addirittura abortita.

Capitolo 3

WiFi

Lo scopo [5] dello standard IEEE 802.11 [16] è quello di fornire connettività wireless a dispositivi o stazioni che richiedono un'istallazione rapida all'interno di una local area. Il suddetto standard definisce le funzioni e i servizi richiesti ai dispositivi compatibili con esso per operare all'interno delle reti ad hoc e con infrastruttura, anch'esse compatibili con lo standard. Esso definisce delle procedure MAC e tecniche di trasmissione dei segnali attraverso un livello fisico (PHY) che potrà essere la radio frequenza (RF) o la radiazione infrarossa (IR). Lo standard descrive anche procedure per fornire un certo livello di riservatezza delle informazioni. Una caratteristica molto importante del protocollo definito da IEEE 802.11 è la trasparenza di comunicazione: nei livelli superiori a quello MAC, un dispositivo 802.11 viene visto come un qualsiasi altro appartenente ad una LAN IEEE 802.x e offre dei servizi comparabili con quelli offerti da quest'ultimo. In altre parole la mobilità dei dispositivi viene gestita a livello MAC.

3.1 Architettura

L'architettura di una rete di comunicazione wireless LAN 802.11 [15,16] si fonda su una struttura a celle, simile a quella dei sistemi di distribuzione per servizi di telefonia GSM (Global System for Mobile communications) dove ciascuna di esse viene chiamata Basic Service Area (BSA). Un gruppo di stazioni situate all'interno di una BSA, in grado di comunicare tra di loro, formano un Basic Service Set (BSS) che viene controllato da una stazione base o Access Point (AP). La maggior parte delle wireless LAN è formata da una molteplicità di celle dove i singoli Access Point sono interconnessi attraverso una qualche tipo di rete di distribuzione, normalmente definita Distribution System (DS). Quello che si ottiene è chiamato Extended Service Area (ESA). La rete di distribuzione è solitamente costituita da una dorsale Ethernet ed in certi casi è wireless essa stessa. Il complesso delle diverse wireless LAN interconnesse (comprendenti differenti celle, i relativi Access Point e il sistema di distribuzione) viene visto come una

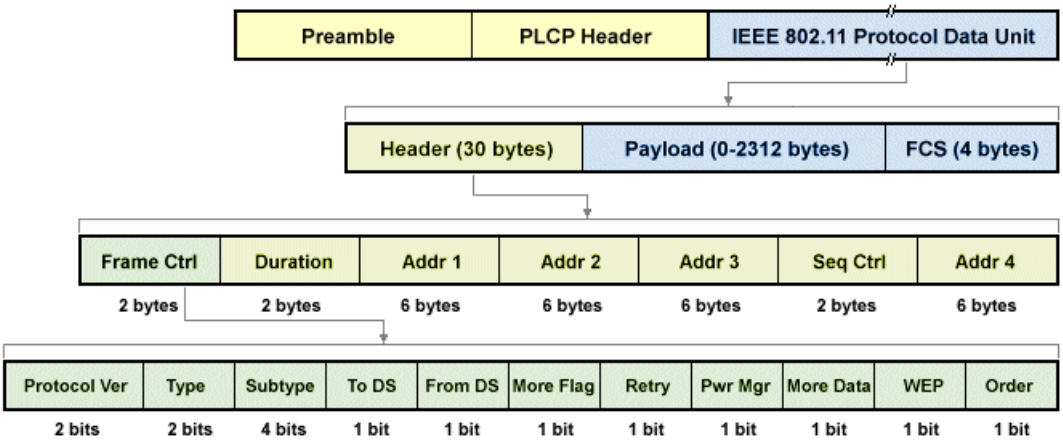

Figura 3.1: Frame IEEE802.11[18]

singola rete 802 dai livelli superiori del modello OSI ed è noto nello standard come Extended Service Set (ESS). La rete può supportare due diversi tipi di wireless LAN.

- Lan Ad Hoc: rete formata da un certo numero di stazioni, contenute in un'area limitata, e caratterizzata da facilità e rapidità di installazione, senza il supporto di una precedente infrastruttura.

- Infrastructure wireless Lan: la rete include nodi speciali, gli Access Point, a ciascuno dei quali compete uno specifico BSS. Gli AP sono fra loro collegati tramite un Distribution System, normalmente in cavo, ma che potrebbe essere anche parzialmente wireless. Il sistema di distribuzione può fornire, inoltre, delle strutture adeguate (Server) per interfacciarsi con reti già esistenti. Lo standard definisce anche il concetto di Portal. Un Portal è un dispositivo che permette l'interconnessione tra una rete LAN 802.11 e un'altra rete 802. Anche se lo standard non lo richiede espressamente, la maggior parte delle installazioni riuniscono l'Access Point e il Portal in un'unica entità fisica.

3.2 I frame 802.11

Il protocollo 802.11 [5] definisce come **frame** il tipo di pacchetto utilizzato sia per la trasmissione dei dati che per il controllo del canale di comunicazione. Ne esistono tre tipi:

- Dati (Data Frame): Usati per la trasmissione dei dati

- Gestione (Management Frame): Servono a scambiarsi informazioni

- Controllo (Control Frame): Gestiscono la trasmissione dei dati

Il primo tipo è quello che si occupa del trasporto vero e proprio dei dati. Gli ultimi due invece servono a creare e gestire il canale, oppure si occupano di gestire la trasmissione dei pacchetti Dati (questi tipi di frame non vengono inoltrati oltre il MAC[1] 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[2] - Inizializzato a 1 se il frame proviene dal sistema di distribuzione.
6. Altri Frammenti (More frag) - Valorizzato con 1 solo se seguono altri frammenti appartenenti allo stesso datagram.
7. Ripetizione (Retry) - Inizializzato a 1 se questo frammento è la ripetizione di un frammento precedentemente trasmesso. Aiuta l'AP nella eliminazione dei frammenti duplicati.
8. Risparmio energia (Pwr mgt) - Inizializzato a 1 se al termine del frame l'interfaccia del mittente entrerà nella modalità di basso consumo. Gli AP non configurano mai questo bit.
9. Altri dati (More Data) - Inizializzato a 1 se il mittente ha altri frame per il dispositivo.
10. WEP – Utilizzato solo se il campo Dati è stato crittografato con l'algoritmo WEP[3].
11. Ordinati (Order) - Se è richiesto il metodo di trasmissione "strict ordering". I frame e i frammenti non sono sempre inviati in ordine perché spesso questo causa rallentamenti nella trasmissione.

3.2.1 I frame di gestione

Tra i frame di gestione segnaliamo:

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

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

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

Authentication Frame Il processo di autenticazione è il meccanismo attraverso il quale un AP accetta o rigetta l'identità di una WNIC[4]. L'interfaccia inizia il processo inviando un Frame di Autenticazione contenente 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 ad un AP quando desidera terminare una comunicazione. L'AP libera la memoria allocata per quella WNIC e rimuove la scheda dalla tabella delle associazioni.

Association Request Frame Un'interfaccia inizia il processo di associazione mandando uno di questi frame all'AP. Il frame contiene informazioni sulla WNIC e l'SSID[5] della rete a cui desidera associarsi. Se l'AP decide di accettare la richiesta, alloca le risorse per la scheda ed invia 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.

[4]WNIC o Wireless Network Interface Controller, sta ad indicare una scheda di rete basata sulla trasmissione radio; essa si pone solitamente ai livelli 1 (fisico) e 2 (datalink) del modello di rete ISO/OSI.

[5]SSID è il nome con il quale una rete wireless si identifica all'esterno, gli Access Point devono annunciare tale informazione con una certa frequenza attraverso i Beacon Frame.

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

3.2.2 I frame di controllo

I frame di controllo, 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 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.

3.3 Strati del protocollo

Lo standard 802.11 è afferente alla famiglia di standard 802.x la quale si occupa di implementare i livelli fisico e data link del modello ISO/OSI. In generale, parlando dello standard 802.11, le specifiche definiscono un singolo livello MAC che può interagire con tre livelli fisici operanti a velocità variabili:

- Frequency Hopping Spread Spectrum (FHSS) nella banda ISM 2,4GHz.
- Direct Sequence Spread Spectrum (DSSS) nella banda ISM 2,4GHz.
- Trasmissione infrarossa (IR).

Oltre alle funzionalità standard usualmente fornite dai livelli di MAC dei protocolli 802.x, il MAC 802.11 supporta delle funzionalità aggiuntive, tipiche dei livelli superiori dello stack protocollare, come la gestione della frammentazione delle Protocol Data Unit, la ritrasmissione dei pacchetti e la gestione dell'acknowledge.

3.4 Livello MAC

Il livello di MAC definisce due differenti metodi di accesso: Distributed Coordination Function [17] e Point Coordination Function [17].

3.4.1 Distributed Coordination Function (CSMA/CA)

Il meccanismo di accesso base si fonda su quello di accesso multiplo con rilevamento della portata e prevenzione delle collisioni (Carrier Sense Multiple Access con Collision Avoidance o, in forma più compatta, CSMA/CA). Quando una stazione vuole trasmettere, testa il canale di trasmissione: nel caso sia occupato (una delle altre stazioni connesse sul medesimo mezzo sta trasmettendo), la stazione deferisce la trasmissione ad un momento successivo. Se invece si rileva che il mezzo è libero, alla stazione è consentito trasmettere. Questi tipi di protocolli sono molto efficienti se il mezzo di trasmissione non è pesantemente caricato, in quanto le stazioni possono trasmettere con il minimo ritardo. Vi è però la possibilità che più stazioni, rilevando contemporaneamente che il mezzo trasmissivo è libero, comincino a trasmettere simultaneamente. Questa situazione, che può creare collisioni sul mezzo, deve essere rilevata in modo che i pacchetti possano essere ritrasmessi direttamente dal livello di MAC in caso di perdita, senza interessare i livelli superiori dello stack protocollare, cosa che produrrebbe ulteriori ritardi. Nel caso dell'Ethernet, questa situazione di collisione è rilevata dalla stazione trasmittente, la quale entra in una fase di ritrasmissione basata su un algoritmo di posticipo della trasmissione denominato Exponential Random Backoff Algorithm. Questo fissa un tempo di ritrasmissione arbitrario al termine del quale viene testato il mezzo trasmissivo e, nel caso sia ancora occupato, viene aumentato esponenzialmente il tempo di ritrasmissione. Il meccanismo, tuttavia, non può essere applicato a comunicazioni su canale radio, principalmente per due motivi:

- il rilevamento della collisione richiederebbe l'immediata implementazione di comunicazione full-duplex; ciò porterebbe ad un significativo incremento del prezzo degli apparati.
- in un ambiente wireless non è possibile assumere che una stazione sia in grado di sentire l'attività di tutte le altre (questa ipotesi è alla base dello schema di rilevamento della collisione). In quest'ottica, se una stazione che vuole trasmettere rileva la non occupazione del mezzo, non necessariamente significa che il mezzo sia libero attorno all'area di ricezione.

Allo scopo di superare questi problemi, il protocollo 802.11 utilizza un meccanismo di collision avoidance unito ad uno schema di positive acknowledge. Una stazione che vuole trasmettere testa il mezzo trasmissivo. Nel caso sia occupato la trasmissione verrà deferita. Al contrario, se è libero e rimane in questo stato per un certo tempo utile denominato Distributed Inter Frame Space(DIFS), la stazione effettua la trasmissione. La stazione ricevente controlla

il CRC del pacchetto ricevuto e risponde con un ACK. Se non viene ricevuto questo segmento, il mittente cercherà di ritrasmettere il pacchetto. E' comunque fissato un numero massimo di ritrasmissioni oltre il quale il pacchetto viene scartato.

3.4.2 Virtual Carrier Sense

Allo scopo di ridurre la probabilità che si verifichi una situazione di collisione tra due stazioni (a causa della impossibilità di ciascuna stazione di coordinarsi con tutte le altre) lo standard definisce un meccanismo denominato Virtual Carrier Sense. Una stazione che vuole trasmettere innanzitutto procede all'invio di un breve pacchetto di controllo denominato RTS (Request To Send), che contiene l'identificativo della sorgente e della destinazione, oltre alla durata della successiva trasmissione di dati. La stazione di destinazione risponde (se il mezzo è libero) con un pacchetto di controllo denominato CTS (Clear To Send), con la stessa informazione relativa alla durata di trasmissione. Tutte le stazioni, ricevendo sia un RTS che un CTS, impostano l'indicatore Virtual Carrier Sense (chiamato NAV che sta per Network Allocation Vector) per un certo tempo, ed utilizzano questa informazione insieme con il Physical Carrier Sense al momento in cui vanno ad effettuare la rilevazione di occupazione del mezzo. Questo meccanismo riduce la probabilità di collisione su un'area di ricezione che è nascosta all'interno dell'intervallo di tempo necessario alla trasmissione dell'RTS, poiché la stazione ascolta il CTS e a causa di ciò vede il mezzo occupato fino alla fine della trasmissione. L'informazione relativa al tempo di trasmissione protegge inoltre l'area del trasmettitore dalle collisioni durante l'ACK da parte di quelle stazioni che sono fuori dall'area di visibilità della stazione che deve fornire la conferma.

3.4.3 Point Coordination Function

Oltre alla funzione base di coordinazione, denominata Distributed Coordination, è prevista una Point Coordination Function, che può essere usata per implementare servizi che hanno necessità di risposta più veloce, come le trasmissioni audio o video. Questa funzione fa uso dell'elevata priorità che l'AP può guadagnare attraverso l'utilizzo di un breve Inter Space Frame (PIFS). Utilizzando questa elevata priorità di accesso, l'AP emette, secondo un meccanismo di polling, delle richieste alle stazioni per la trasmissione dati, quindi controlla l'accesso al mezzo. Allo scopo di consentire alle stazioni regolari di accedere al mezzo trasmissivo, è prevista la norma in base alla quale l'AP deve lasciare abbastanza tempo per il Distributed Access all'interno della PCF.

3.4.4 Metodi di accesso ad una BSS

Nell'ambito wireless il processo di scoperta ed identificazione della rete è detto scanning. I diversi parametri che sono utilizzati in questa procedura possono essere specificati dall'utente ma molti sono di default presenti nei driver delle schede wireless. Nello scanning passivo la stazione rimane in ascolto dei pacchetti di beacon dell'Access Point che contengono i parametri necessari ad una stazione al fine di collegarsi al network: tipo di BSS, SSID, timer per la sincronizzazione e canale da utilizzare. Nello scanning attivo una stazione, piuttosto che ascoltare, trasmette dei pacchetti canale per canale, richiedendo l'accesso ad uno specico BSS. Al termine dello scan, attivo o passivo, la stazione ha l'insieme dei parametri necessari a connettersi alla rete: tipo di BSS, SSID e canale da utilizzare. A questo punto la stazione può passare all'autenticazione. Nell' 802.11 utilizza due metodi di autenticazione:

- Open System: l'AP accetta la connessione da qualunque stazione senza verificarne l'identità. L'unico parametro identificativo preso in considerazione è il MAC[6] address sul quale è possibile eseguire delle regole di filtraggio.
- Shared Key: viene utilizzato l'autenticazione di tipo WEP (Wired Equivalent Privacy) ed implica che ogni stazione abbia il WEP attivo ed utilizzi una shared key. In questa procedura viene introdotto un meccanismo di controllo della shared key tramite l'utilizzo della stessa per criptare un challenge text e la sua decriptazione che può avvenire solo se le due shared key sono uguali.

Una volta che l'autenticazione è stata completata la stazione può associarsi ad un AP: questa è una procedura di registrazione della identità della stazione in modo che il sistema di distribuzione sappia sempre dove la stazione si trova per indirizzare correttamente i frame. Dopo che l'associazione è stata completata, l'AP deve registrare la stazione nella rete cablata in modo che i frame destinatogli siano correttamente ricevuti. Un metodo di registrazione consiste nell'associare il MAC della stazione alla porta dello switch dove l'AP è collegato tramite il protocollo ARP. Dopo che la stazione è autenticata sull'AP viene inviato un pacchetto di association request. Se l'elaborazione di questo segmento termina con successo è possibile l'inizio della trasmissione dei dati per la stazione.

3.5 Livello Fisico

Il livello fisico è composto da due sottolivelli denominati Protocollo di Convergenza del livello fisico (PLCP) e il sistema dipendente dal mezzo (PMD). Il livello PMD si interfaccia direttamente col mezzo fisico che consiste nella trasmissione di un segnale a Radio Frequenze o ad

[6]Si parla qui di MAC, inteso come indirizzo univoco di una scheda di rete (Media Access Control) formato da 6 byte; si riferisce quindi ad un aspetto hardware, ma è solitamente possibile modificarlo via software.

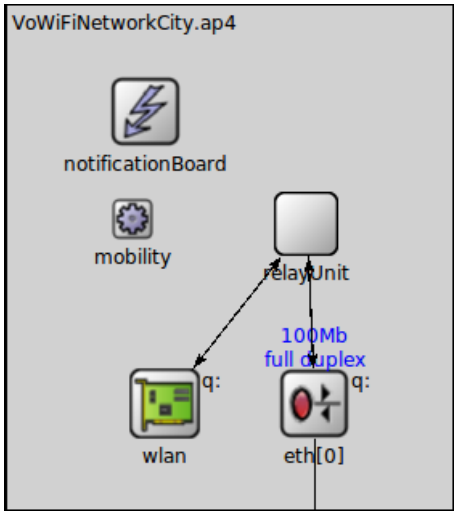

Figura 3.2: Architettura di un Access Point in INET

Infrarossi. Il sottolivello PMD riceve in ingresso uno stream di dati dal PLCP e lo converte in un formato che possa essere trasmesso ad un'altra stazione. Il sottolivello PLCP è progettato per convertire i dati in uscita dal sottolivello MAC in un formato che possa essere trasmesso dal sottolivello PMD. In generale, ogni configurazione del livello PMD richiede uno specifico PLCP, permettendo al livello MAC di mantenersi indipendente dal livello PHY. Il PMD, nel protocollo 802.11, permette l'utilizzo di tre diversi progetti di gestione del livello fisico: il Frequency Hopping Spread Spectrum (FHSS), il Direct Sequence Spread Spectrum e la trasmissione ad Infrarossi.

3.6 WiFi in INET/OMNET++

In INET[7] ogni nodo di rete di tipo wireless è un aggregato di moduli, dove è sempre presente il modulo *Ieee80211NicAP* di livello *linklayer Ieee802.11*. Una interfaccia WNIC (Ieee80211) nel framework INET è composta dai seguenti quattro moduli composti tra loro:

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

[7]INET è un framework modulare di OMNET++ per la creazione e gestione di reti, nodi e protocolli.

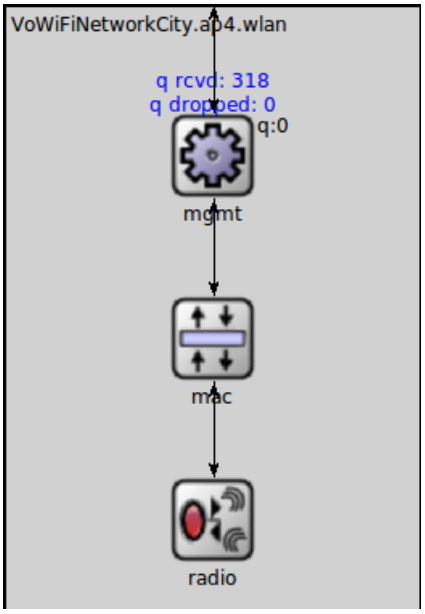

Figura 3.3: Architettura di una WNIC in INET

L'agent L'agent [5] è il modulo che si occupa di gestire il comportamento del livello a lui annesso (management). Si occupa di ordinare (attraverso messaggi command) a quest'ultimo di condurre operazioni quali la scansione dei canali radio, l'autenticazione o l'associazione con un Access Point. Il livello management si limita ad eseguire questi comandi per poi riportare il risultato all'agent. Modificando o rimpiazzando un agent, si può modificare il comportamento stesso di uno STA ed implementare algoritmi o strategie necessari alla propria simulazione.

Il manager Il manager [5] 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. In INET [12] questo modulo non modifica mai il canale di comunicazione, ma lavora su quello che il livello fisico ha configurato per la scheda nel modulo *Ieee80211Radio*.

Il livello MAC In INET [12], assieme al modulo Radio, il modulo MAC costituisce il livello fisico, quindi l'incapsulamento e il decapsulamento vengono fatti nel Manager, il modulo si aspetta in input da quest'ultimo dei frame di tipo *Ieee80211DataOrMgmtFrame*, sottoclasse dei frame *Ieee80211Frame*, dei quali setta solo i campi "Transmitter Address", "Frame Sequence" e "Fragment Number", tutto il resto viene riempito nel livello soprastante quando parte la trasmissione e quindi la discesa del frame a questo livello.

Non sono supportati da INET la frammentazione, il power management e il polling[8], quindi tutti i campi del ManagementFrame riguardanti questi aspetti sono stati omessi dall'implementazione.

Il livello fisico Il livello fisico [5] 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 livello MAC.
Nel caso di INET [12], il modulo *Ieee80211RadioInterfDist* gestisce il livello fisico Ieee802.11, tale modulo è una sottoclasse di *Radio*, prototipo per tutti i moduli radio presenti nel framework. Il livello gestisce la trasmissione e la ricezione dei frame, incapsulandoli (o decapsulandoli) in ulteriori pacchetti che virtualizzano i segnali radio in uscita per le altre station, essi avranno appositi campi per specificare la potenza del segnale e quant'altro. Questo livello si occupa inoltre del bitrate e del canale di comunicazione, che qualora venga modificato, si attende innanzitutto la fine della trasmissione in corso ed in seguito notificato ai livelli soprastanti.

[8]Il PCF, Point Cordination Function è una tecnica di Medium Access Control per gestire i conflitti tra stazioni wireless e non si serve di frame come il meccanismo RTS/CTS.

Capitolo 4

TCP a Ritrasmissione Anticipata sul Nodo Mobile

Come si spiegherà in seguito, lo scopo della tesi è creare un meccanismo di ritrasmissione anticipata su un Access Point. Lo scopo è di ritrasmettere a livello datalink (anticipando i timer TCP) quei frame, che per fattori essenzialmente "ambientali", non raggiungono la station destinazione nel raggio di trasmissione dell'AP.
Il lavoro di Paolo Alberti [6] consiste nell'applicare il sistema RWMA anche al protocollo TCP, cosa che fino ad ora riguardava solo il protocollo UDP; la motivazione principale per cui si vuole effettuare la modifica al protocollo è la possibilità di perdite dovute a cause esterne durante le comunicazioni wireless.
In questo scenario, fattori ambientali, rumori e ostacoli possono influire sulla comunicazione con l'Access Point, riducendo effettivamente la QoS. Test svolti mostrano che una semplice comunicazione in cui l'host mittente si muove in un ambiente domestico, può portare a perdite di pacchetti, senza che l'interfaccia di rete smetta di funzionare a causa dell'eccessiva lontananza dall'AP; si noti che allontanandosi dall'AP è possibile che si presentino delle perdite senza che comunque l'interfaccia di rete perda completamente la connessione. Scopo della nostra modifica sarà la gestione anticipata di queste perdite, riscontrandole prima dello scadere del timeout di TCP.

Progettazione Per la realizzazione della modifica [6], il principio che si vuole sfruttare è il passaggio di informazioni sull'errore di invio da livello MAC a quelli trasporto. Modificando un sistema con questo ulteriore controllo, sarebbe teoricamente possibile aumentare la QoS della trasmissione su interfaccia wireless; riuscendo, quindi, a capire se il pacchetto è arrivato con successo all'AP a cui il nodo mobile è connesso o se quest'ultimo l'ha scartato per problemi di congestione, è possibile una gestione più veloce dell'errore. La struttura base del codice sareb-

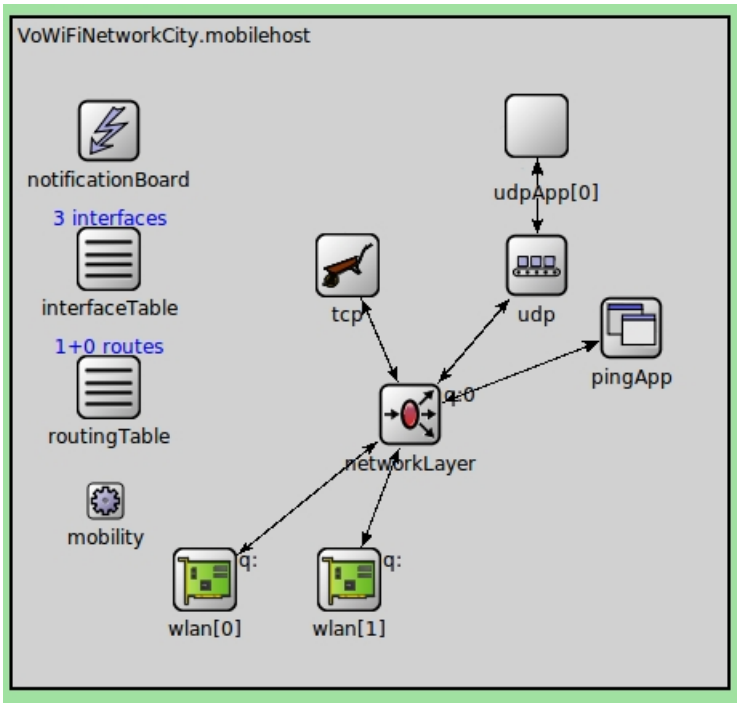

Figura 4.1: Il nodo mobile originario

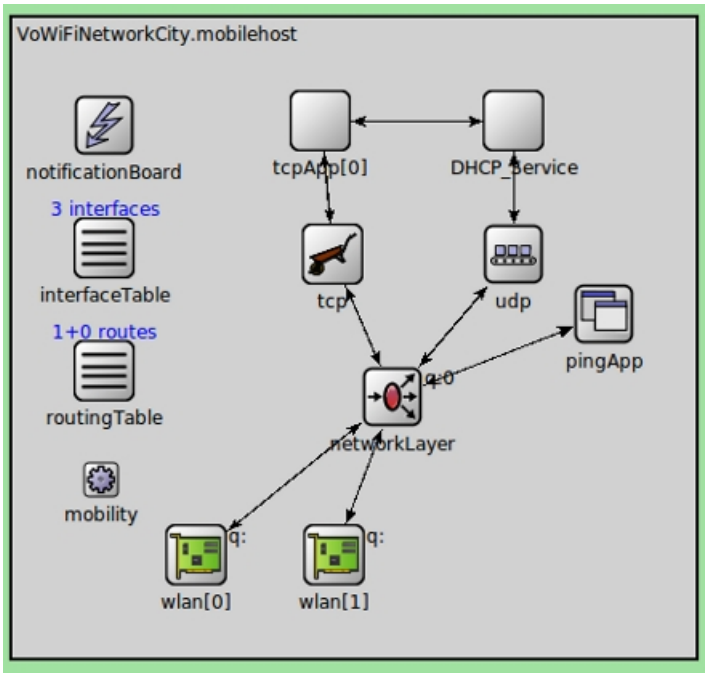

Figura 4.2: Il nuovo nodo mobile

be quella simile a quella per UDP[1], con la principale differenza che, mentre con UDP il modulo ULB di RWMA si trova a livello applicazione e quindi è in questo strato che vengono effettuate tutte le decisioni riguardanti le ritrasmissioni del pacchetto in caso di non arrivo al primo hop, con TCP, il modulo ULB viene integrato all'interno del modulo TCP, lasciando l'applicazione TCP ignara su cosa sta accadendo a livello transporto e lasciando a quest'ultimo strato il compito di ritrasmettere il pacchetto. A questo, va aggiunta la difficoltà della ritrasmissione basata sui timer, come visto per le versioni standard del protocollo TCP.

[1]In RWMA il meccanismo di QoS implementato riguarda il protocollo UDP. Le modifiche principali si concentrano sul nodo mobile, dove il livello applicativo (l'ULB) gestisce la ritrasmissione e la gestione dei pacchetti, un modulo a livello datalink (TED) riscontra la presenza di errori/perdite, infine un'ulteriore applicazione (Monitor) gestisce e configura le interfacce wireless. Lo scopo di questo meccanismo è aumentare la QoS in quelle connessioni che prefigurano una continuità e stabilità della connessione su UDP, ad esempio il Voice Over IP (VOIP).

Capitolo 5

Obiettivi

Presentiamo in questo capitolo il concetto fondante del meccanismo implementato per specificarne lo scopo, contestualizzandolo nell'ambiente di lavoro utilizzato.

Scenario L'enviroment nel quale si situa il progetto è simulato sulla piattaforma OMNET++ con framework INET, nella fattispecie si tratta di un comune scenario cittadino, dove sono presenti Station, Router, Access Point e vari ostacoli alla trasmissione di questi ultimi, i quali forniscono accesso alla rete a tutti quei dispositivi dotati di connettività wireless.
Prendiamo in considerazione ora una normale comunicazione basata su connessione TCP tra un client e un server ad una certa distanza l'uno dall'altro; il client TCP è un processo che gira su una station connessa ad un Access Point Wireless, mentre il server corrispondente è attivo su una station in una rete differente, collegata con la precedente tramite cablatura.
Supponiamo innanzitutto una certa distanza (quindi abbastanza elevato RTT[1]) tra client e server, non essendo comunque un'ipotesi vincolante, e in secondo luogo supponiamo che la sottorete wireless del client sia soggetta ad errori e perdite di pacchetti nella comunicazione radio; anche quest'ultimo aspetto sarà parte della parte progettuale/implementativa della tesi.
Le problematiche che si possono riscontrare in una situazione del genere sono di semplice intuizione, analizziamo i seguenti aspetti.

- Nelle reti wireless, l'ambiente esterno è spesso fonte di perdite ed errori, come già visto, causati [13] da fattori come: dispositivi che emettono onde radio in uno spettro paragonabile a quello imposto da IEEE802.11, presenza di persone ed oggetti nello spazio di copertura dell'AP, distanza eccessiva tra station e AP, e così via.

- La distanza fra client e server, per come l'abbiamo supposta è notevole, ciò implica che TCP calcolerà i suoi timer in base a questa informazione, soprattutto quello riguardar-

[1]Round Trip Time, ovvero il tempo che un pacchetto impiega per viaggiare da un computer ad un altro, complessivo di andata e ritorno.

dante la ritrasmissione del pacchetto perso e/o corrotto, quindi in base al meccanismo di ACK.

- Ogni volta che un pacchetto viene perso nell'ultimo hop (tra Access Point e Station Wireless) verrà ritrasmesso per le ulteriori 6 volte imposte dallo standard; quando le *chance* del frame sono esaurite, quest'ultimo viene scartato nell'Access Point e fin quando il timer e le finestre TCP non lo richiedano, il frame non raggiungerà il client.

Idea generale La filosofia che sta dietro alla tesi è quella di voler migliorare la Quality Of Service del modello a partire da queste semplici considerazioni. Ciò che si intende fare è, infatti, evitare di dover richiedere un frame a chilometri di distanza situato sul server mediante timeout TCP, quando lo si aveva a disposizione a qualche metro sull'Access Point, che invece per protocollo ha dovuto scartarlo, avendolo ritrasmesso per un numero limite di volte.
L'idea, in pratica, è quella di rendersi conto sull'Access Point di quando si ha innanzitutto a che fare con una comunicazione TCP transitante nello stack di rete, in seguito si vuole implementare un qualche meccanismo che ci permetta di salvaguardare il pacchetto che andrebbe scartato e "regalargli" una chance ulteriore di arrivare a destinazione.

Ambito di applicazione Nello specifico verrà simulata l'attività di alcuni nodi di rete sia fissi che mobili che comunicheranno tra di loro e all'esterno della loro sottorete, nello specifico con una rete a distanza più elevata per testare connessioni con più alto Round Trip Time.
Ci si concentrerà sull'attività specifica di un nodo mobile dotato di più interfacce di reti, il quale si sposterà attraverso lo scenario cittadino ed arrivato in corrispondenza di uno specifico Access Point inizierà ad inviare e ricevere dati da un host (il Serverproxy) situato nella rete a distanza mediante connessione TCP con il pattern Client-Server. Questo particolare frangente sarà oggetto di studio e di implementazione del meccanismo di QoS.

Ritrasmissione anticipata su Access Point Il concetto principale della modifica qui introdotta è esattamente lo stesso che si è implementato sul lato client, la differenza è che ora si vuole applicarlo ad un Access Point, che si differenzia su INET per l'esistenza di due soli livelli di rete, quindi si lavorerà essenzialmente sul sottolivello MAC.
Il problema che si va ad affrontare nasce dalla semplice domanda: *è sensato ritrasmettere un pacchetto perso nell'ultimo hop nella trasmissione radio da AP a Station, prima dello scadere del timer TCP, e quindi del reinvio dall'endpoint?*
Essenzialmente ciò che si vuole realizzare è un meccanismo che riconosca la perdita di un pacchetto (se TCP) nell'ultimo hop, e dargli un'ulteriore chance di ritrasmissione (prioritariamente rispetto agli altri pacchetti in coda) diminuendo il carico sulla rete.
Ci si aspettano i seguenti risvolti positivi, se il meccanismo funzionasse correttamente:

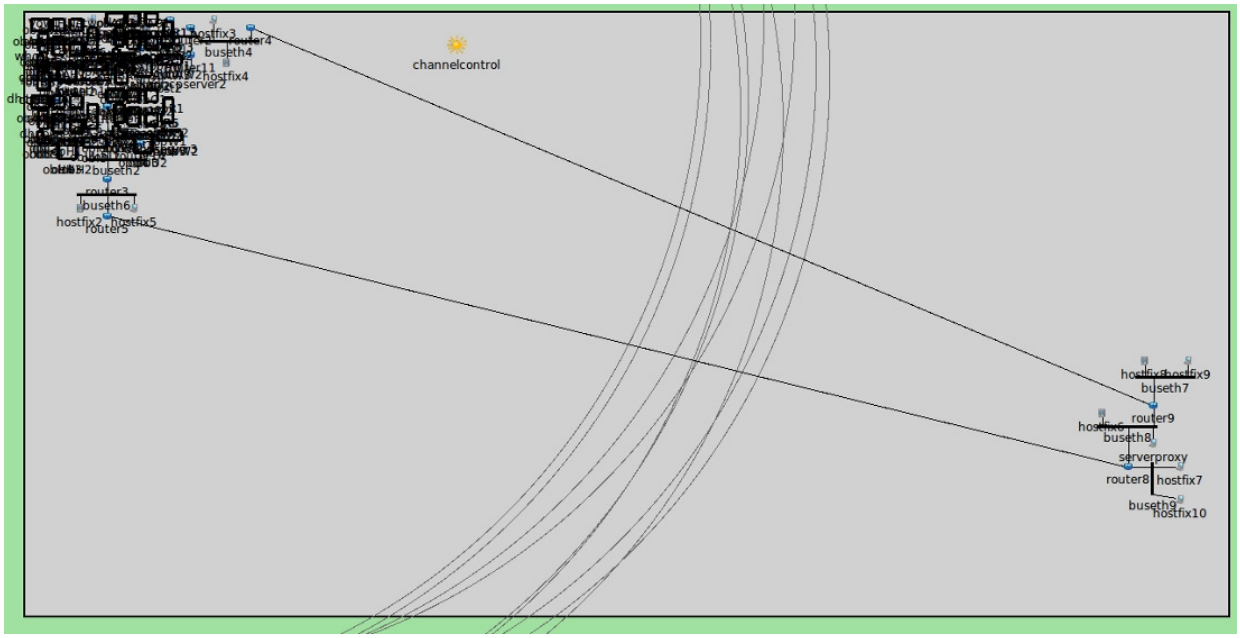

Figura 5.1: Scenario simulazione in INET/OMNET++

1. la velocizzazione della ricezione da parte della station quando vi è un pacchetto perso “negli ultimi metri”, senza dover aspettare il reinvio dall’end-system corrispondente allo scadere del timer TCP.

2. come diretta conseguenza del primo punto, si potrebbe ottenere una minore congestione della rete esterna che non viene appesantita da reinvii TCP, dovuti a problemi (interferenze di segnale etc.) esistenti nella rete wireless.

Eventuali ulteriori vantaggi e/o svantaggi del meccanismo sono rimandati ed analizzati in seguito ai risultati dei test svolti.

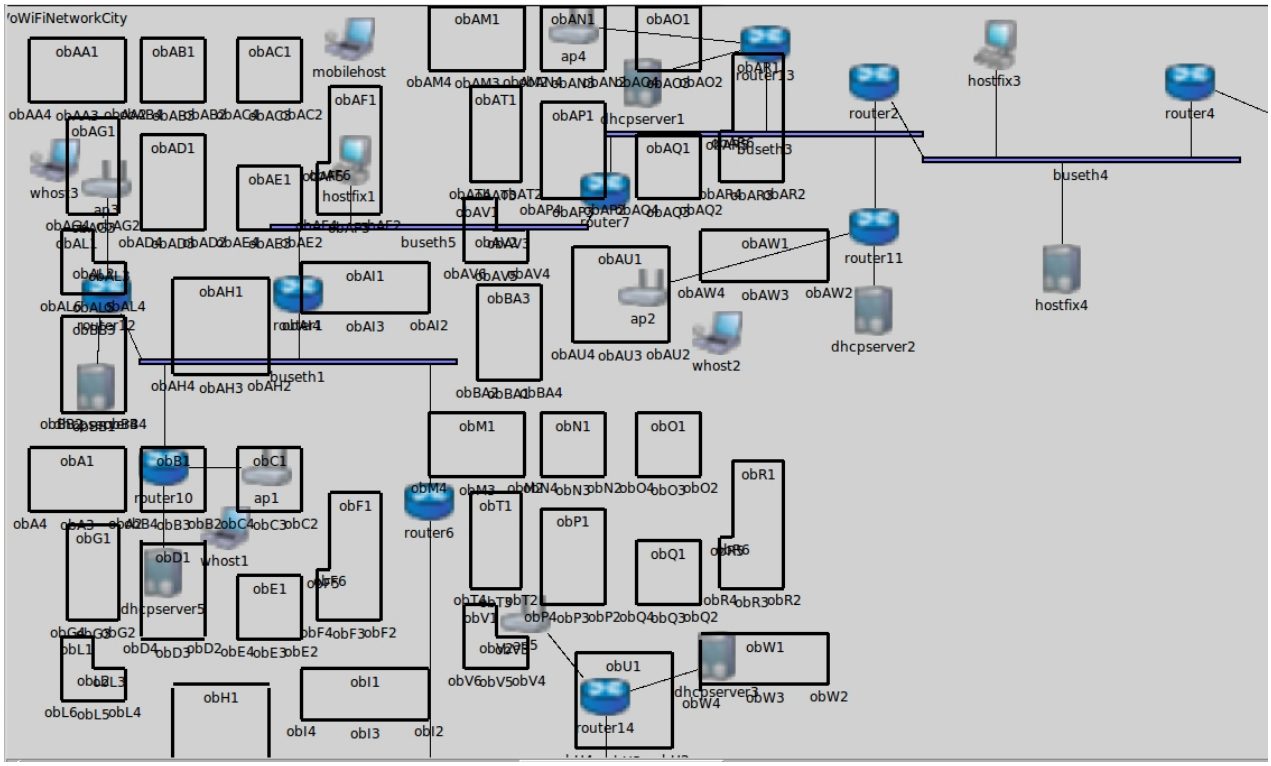

Figura 5.2: Dettaglio scenario simulazione in INET/OMNET++

Capitolo 6

Progettazione

Ciò che si vuole realizzare è l'applicazione di RWMA alle connessioni TCP lato Access Point, finora presente per UDP e per TCP solo sul lato Host, migliorando la risoluzione al problema della perdita di pacchetti in una rete wireless causate da fattori esterni; lo scopo è la gestione di queste perdite anticipatamente rispetto a quanto TCP farebbe da protocollo con l'uso dei timeout.
Il primo passo è sicuramente quindi quello di individuare le perdite, in un secondo momento analizzare come gestirle e poi valutare le azioni da intraprendere.

La modifica riguarderà soprattutto il sottolivello MAC dell'Access Point, si modificherà poi il comportamento del proxy server aggiungendo un'applicazione server TCP, verrà modificato il sottolivello MAC del nodo mobile e l'applicazione client TCP connessa al serverproxy; infine alcune ulteriori piccole modifiche saranno utili per la fase di testing e di logging del sistema sviluppato.
Nello specifico si interverrà sui seguenti moduli.

Access Point Il cuore della modifica agisce sul sottolivello MAC dell'AP, distinguendo le seguenti fasi di implementazione.

- Riconoscimento della perdita: il primo passo consiste nell'individuare il pacchetto che, ritrasmesso per 7 volte (da protocollo IEEE 802.11), verrebbe definitivamente eliminato dalla coda di trasmissione dell'Access Point. Questo in particolare avviene qualora non venga ricevuto per 7 volte il pacchetto ACK dall'Host in seguito all'invio del frame, segnale sicuro di perdita dello stesso.
  Per effettuare questa modifica si è introdotto del codice nel modulo *Ieee80211Mac*, individuando quando la FSMA[1] si trova nello stato di WAITACK e, contemporaneamente, il

[1] FSMA è l'implementazione di una macchina a stati finiti, con la quale il modulo MAC gestisce gli stati in cui si trova, adoperando funzioni diverse a seconda di questi.

contatore di ritrasmissione indica che non vi sono più chance per la trasmissione corrente. A questo punto si analizza il messaggio corrente, come specificato in seguito, e nel caso in cui la ritrasmissione è possibile, questa viene subito ordinata alla FSMA, scavalcando prioritariamente le altre, dando quindi una possibilità ulteriore ed immediata al frame di essere ricevuto dalla station.

- Analisi del pacchetto perso: innanzitutto, ogni qualvolta un pacchetto arriva nel modulo *Ieee80211Mac* alla fase di trasmissione e quindi di ricezione dell'ACK corrispondente, viene decapsulato, per verificare che si tratti effettivamente di un pacchetto TCP (o TCP_RWMA). Se il frame risponde alla nostre esigenze, si estrae un identificatore (il SequenceNumber del frame 802.11) che viene salvato in un'apposita lista di ritrasmissione, in aggiunta all'ulteriore contatore sui rinvii da effettuare, nel caso in cui sviluppi futuri implichino un maggior numero di reinvii possibili.

- Ritrasmissione: come accennato precedentemente, a questo punto, si decide se il pacchetto è da ritrasmettere immediatamente (regalando altri 7 tentativi di ritrasmissione a livello MAC), oppure se è da scartare definitivamente (ad esempio se già ha beneficiato del meccanismo), lasciando il compito al soprastante livello TCP e ai suoi timer la risoluzione alla perdita. La scelta viene effettuata in base al contatore nella struttura associata al frame, o nel caso di una singola ritrasmissione possibile, si evince se il pacchetto è già stato ritrasmesso se il suo identificatore è già presente nella lista, ovvero ha già subito il reinvio ed è stato inserito in coda. Ogni volta che un pacchetto non riesce, nonostante il meccanismo, ad essere inviato, viene eliminato assieme alla struttura che lo riguarda nella lista di ritrasmissione.

- Logging: questa fase è parallela a tutto il meccanismo, ed è utile al testing e al logging del sistema, vi sono essenzialmente dei contatori che monitorano alcune variabili, come il numero di pacchetti inviati e così via.

Server Proxy Al nodo serverproxy viene semplicemente caricato un modulo ulteriore di tipo *tcpapp*, che nel caso specifico è un server TCP, il quale alla ricezione di un messaggio di tipo *GenericAppMsg*, con intestazione "WF-X" (con X, variabile incrementale), creato dall'applicazione client, lo rispedisce al mittente modificando l'intestazione in "RPYWF-X".

Mobile Node In questo nodo vi è caricato il client TCP, realizzato dalla classe *TCPAppRWMA*, che essenzialmente crea un socket, effettua una bind, e dopo aver ottenuto l'indirizzo IP locale dal DHCP Service esegue una connect sul socket.
Una volta attiva la connessione, l'applicazione invia periodicamente un messaggio, con intestazione "WF-X", al serverproxy, specificando la lunghezza della risposta che ci si aspetta dal

server; quando l'applicazione TCP riceve il pacchetto IPComunication in cui lo stato dell'interfaccia che sta utilizzando è DISABLED, allora viene richiamata la funzione close sul socket e terminata la simulazione.
La modifica effettuata in questo modulo, oltre all'intestazione del messaggio, riguarda il sottolivello (già modificato in RWMA) *MacRWMA*, dove si è implementato un meccanismo per simulare la perdita di pacchetti nell'ultimo hop, ricevuti quindi dall'Access Point modificato. Quello che si è progettato è la randomizzazione (con coefficiente di perdita modificabile) del drop dei frame 802.11 in arrivo dal livello fisico, selezionando solo quelli da noi monitorati.
Come negli altri moduli, è stato inoltre inserito un meccanismo di logging per la fase di test.

FiberLine Questa classe di INET, estensione del Datarate Channel è stata modificata per individuare e contare i pacchetti che passano nella linea cablata tra la sottorete del serverproxy e quella da cui trasmette il nodo mobile, al fine di effettuare alcune statistiche sulla convenienza del meccanismo sviluppato.

File di configurazione INET/OMNET++ Alcune modifiche ai file strettamente riguardanti OMNET sono state fatte per ridurre il traffico nella sottorete del nodo mobile, al solo fine di facilitare il testing. Altre modifiche sono invece eseguite per modificare i parametri della simulazione e del testing.

6.1 Alternative Progettuali

Inizialmente fu concepita una diversa possibile modifica ai moduli trattati, soprattutto al livello MAC dell'Access Point, come segue. Innanzitutto, così come la modifica effettuata, sarebbero stati decapsulati e riconosciuti quei pacchetti TCP ritrasmessi fino al punto precedente all'eliminazione dal livello MAC, poi si sarebbe fatta una copia del pacchetto e delle informazioni necessarie, salvandole in un'opportuna lista di strutture. Si sarebbe poi trattato il reinvio dei pacchetti, in ordine di arrivo nella lista, con un eventuale timer allo scadere del quale sarebbe partita la ritrasmissione. Si sarebbe prevista infine una eventuale gestione dei reinvii in base ad un timestamp, salvato assieme al frame, per evitare di ritrasmettere pacchetti persi ma con molta probabilità già rispediti dall'endpoint allo scadere del timer TCP.

Capitolo 7

Note Implementative

Il sistema è stato sviluppato su kernel Linux Linux 3.10.1, inoltre si utilizza il simulatore OMNET++ nella versione 4.1 congiuntamente al framework INET compatibile con esso.

7.1 File

Per l'implementazione del sistema sono stati modificati i seguenti file (a partire dalla directory principale di INET):

- FiberLine.cc
- FiberLine.h

examples/wireless/voiceoverwifiRWMA/:

- rwmaMonitorCity.ini

src/applications/tcpapp/:

- TCPAppRWMA.cc
- TCPAppRWMA.h
- TCPGenericSrvApp.cc
- TCPGenericSrvApp.h
- TCPSrvHostApp.cc
- TCPSrvHostApp.h

src/linklayer/ieee80211/mac/:

- Ieee80211Mac.cc
- Ieee80211Mac.h
- Ieee80211MacRWMA.cc
- Ieee80211MacRWMA.h

7.2 Compilazione ed esecuzione

Se si vuole compilare il codice a seguito della creazione di nuovi file, bisogna invocare il comando **make -f makemakefiles** nella directory principale di INET, che aggiornerà il makefile, e poi invocare lo script **makeandcopy**.

Se invece si intende compilare il codice a seguito di sole modifiche a file preesistenti, basterà eseguire appunto lo script **makeandcopy** situato nella directory principale di INET, esso si occuperà automaticamente di compilare il codice e copiare l'eseguibile nella cartella di destinazione.

Un possibile problema [6] 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 o file ini, non è necessario ricompilare, ma basta lanciare nuovamente la simulazione. La simulazione creata per testare il sistema, si trova nella cartella /examples/wireless/voiceoverwifiRWMA, nel file *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 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>**

Capitolo 8

Valutazione

8.1 Criteri

Raccogliamo in questo capitolo alcuni test svolti per verificare ed analizzare il funzionamento del meccanismo implementato.
Innanzitutto si descriverà brevemente lo scenario della simulazione, la configurazione nei file adibiti ad essa, ed in generale cosa avviene nel suo svolgimento. In secondo luogo verranno presentati i concetti sui quali sono state prese le misurazioni ed i parametri che si sono presi in considerazione; tra essi sono compresi quelli che vengono fatti variare al fine di testare il meccanismo in svariate condizioni. Queste variazioni dei parametri saranno utili per effettuare poi un'analisi sul funzionamento in generale ma soprattutto con lo scopo di capire quando la ritrasmissione anticipata può effettivamente avere un effetto positivo sulla Quality of Service della connessione TCP ed infine se influisce positivamente anche esternamente ad essa.

8.2 Scenario della Sperimentazione

Per la simulazione [5] viene utilizzato il file *rwmaMonitorCity.ini* (../examples/wireless/voiceoverwifiRWMA), in cui è definita la configurazione NetworkCity; questa fa riferimento alla rete definita in *VoWiFiNetworkCity.ned*.
Lo scenario NetworkCity è costituito dalle due reti in cui sono presenti i diversi dispositivi di rete; il nodo wireless mobile si muove lungo un percorso predefinito passando tra un certo numero di ostacoli, che rappresentano gli edifici in un contesto urbano. All'interno della stessa rete dell'host mobile sono, inoltre, presenti cinque access point collocati in modo tale da fornire una copertura disomogenea per far sì che il nodo mobile, spostandosi fra le aree con intensità di segnale diverse, possa ricevere delle notifiche di mancata trasmissione al primo hop o debba riconfigurare più volte l'interfaccia di rete, provocando così una perdita ulteriore di pacchetti, che dovranno quindi essere ritrasmessi (RWMA trasmissione anticipata di Paolo Alberti).

Quando il nodo mobile arriva in corrispondenza del quarto access point della sottorete urbana si connette al serverproxy nell'altra rete ed inizia a trasmettere, facendo partire di conseguenza il meccanismo di QoS sull'AP, la simulazione di perdita sull'host e tutti i meccanismi di log che serviranno per la fase di test.

Qui di seguito viene presentata parte della configurazione utilizzata, considerando solo i nodi principali della comunicazione:

- Host Mobile:
  - Indirizzo MAC schede di rete: 0A:AA:00:01:00:01 e 0A:AA:00:01:00:02
    - Applicazione: DHCP (sottoclasse di UDPBasicAppForMultipleNics)
    - Indirizzo IP temporaneo: 64.64.64.64
    - Indirizzo IP del server DHCP: 128.128.128.128
    - Porta locale: 80
    - Porta destinazione: 80

    Oltre all'applicazione DHCP è presente l'applicazione TCP:
    - Applicazione: TCPAppRWMA
    - Dimensione pacchetti: 1024B
    - Frequenza pacchetti: 40ms
    - Dimensione pacchetti di risposta: 100B
    - Algoritmo TCP utilizzato: variabile tra TCPReno e TCPRenoRWMA
    - Indirizzo IP destinazione: 128.97.1.131
    - Porta locale: 90
    - Porta destinazione: 90
- Proxy Server:
  - Applicazione: TCPGenericSrvApp
  - Indirizzo IP: 128.97.1.131
  - Porta locale: 90
- Server DHCP:
  - Applicazione: UDPBasicAppForServerProxy
  - Indirizzo IP del client: 64.64.64.64
  - Porta locale: 80

- 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

8.3 Misurazioni ed Analisi

Si sono raccolte alcune rilevazioni in seguito alle molteplici simulazioni svolte, tutte le misurazioni sono riferite in rapporto a 100 frame ricevuti **effettivamente** dal livello TCP del Mobile Node, rispettivamente queste sono:

1. **Frame inviati dal Serverproxy**: il numero di frame rispetto a 100 effettivamente ricevuti dal Mobile Node a livello TCP, inviati dal serverproxy in seguito alla ricezione di un *GenericAppMsg* dal client, più quelli reinviati a causa dello scadere del timeout TCP senza ricezione dell'ACK corrispondente.

2. **Frame persi al livello MAC (Dropped)**: il numero di frame rispetto a 100 effettivamente ricevuti dal Mobile Node a livello TCP, che sono stati scartati nel suo livello MAC, simulando le interferenze ambientali.

3. **Frame inviati dall'Access Point**: il numero di frame rispetto a 100 effettivamente ricevuti dal Mobile Node a livello TCP, che sono inviati dall'Access Point, quindi sono compresi tutti i tentativi, sia quelli dovuti allo standard *Ieee802.11*, sia le ritrasmissioni del meccanismo di QoS sviluppato.

4. **Ritrasmissioni anticipate AP (RWMA)**: il numero di frame rispetto a 100 effettivamente ricevuti dal Mobile Node a livello TCP, che l'Access Point ha inviato in seguito alla decisione di ritrasmissione anticipata. Questo parametro quindi rappresenta l'efficienza del meccanismo di QoS, indicando quanto l'Access Point modificato vada ad evitare le ritrasmissioni dal Serverproxy, per l'appunto, anticipandolo.

Vengono utilizzati i seguenti parametri di configurazione:

- il datarate del cavo di comunicazione tra le reti viene posto a 10Mbps.
- il pacchetto generato dall'applicazione del nodo mobile ha dimensioni pari a 1kB ed è creato ogni 40 ms.
- la latenza del cavo di comunicazione tra le reti può assumere i valori 0ms, 50ms, 150ms oppure 300ms.
- la perdita random del pacchetto a livello di AP può variare tra 65%, 75%, 85%

I primi due parametri sono comuni a tutti i test, mentre gli altri due variano da test a test.

Dati I risultati sono espressi in 3 tabelle in base alla percentuale di pacchetti persi (scartati dal livello MAC del Nodo Mobile). Ulteriormente, all'interno di ciascuna tabella, si dividono i risultati secondo quattro differenti delay di risposta del serverproxy impostati nel file *rwmaMonitorCity.ini*.

Pacchetti persi = 65%

Delay	**Frame inviati dal Serverproxy**	**Frame persi al livello MAC (Dropped)**	**Frame inviati dall'Access Point**	**Ritrasmissioni anticipate AP (RWMA)**
0ms	106	255	287	11
50ms	102	240	261	8
150ms	101	235	260	6
300ms	104	251	282	8

Pacchetti persi = 75%

Delay	**Frame inviati dal Serverproxy**	**Frame persi al livello MAC (Dropped)**	**Frame inviati dall'Access Point**	**Ritrasmissioni anticipate AP (RWMA)**
0ms	114	376	397	16
50ms	106	357	377	17
150ms	107	339	371	15
300ms	106	359	377	17

Pacchetti persi = 85%

Delay	**Frame inviati dal Serverproxy**	**Frame persi al livello MAC (Dropped)**	**Frame inviati dall'Access Point**	**Ritrasmissioni anticipate AP (RWMA)**
0ms	128	656	657	40
50ms	130	641	644	36
150ms	137	664	662	39
300ms	129	659	660	38

Analisi dei dati Come si può notare dai dati, innanzitutto, l'influenza maggiore sul lavoro ulteriore del server non è data tanto dalla latenza, quanto con maggior gravità dalla perdita di pacchetti, come ci si può aspettare; si nota infatti che all'aumentare della percentuale di pacchetti *dropped* aumenta il lavoro del server, quindi dei pacchetti ritrasmessi.
In secondo luogo, e questo è il dato di maggior rilievo, si può notare come il meccanismo implementato aiuti di molto il server, arrivando ad anticipare fino al 40% dei pacchetti trasmessi; si consideri che questi frame sono quelli reinviati dall'Access Point con la modifica effettuata, questo implica che:

1. si è diminuito di una misura considerevole il numero di ritrasmissioni che deve effettuare il server.

2. si riduce il tempo di attesa del client, dando una possibilità ulteriore ai pacchetti persi nell'ultimo hop.

3. si riduce il numero di pacchetti che viaggiano al di fuori della sottorete del client, non sovraccaricando la rete esterna di frame che andrebbero scartati dall'Access Point e che invece possono essere ritrasmessi "internamente".

Si notino le differenze riguardanti le ritrasmissioni anticipate a seconda dei due parametri variabili.
Si vede chiaramente che il meccanismo dà il suo meglio laddove la perdita di pacchetti nella rete wireless sia molto alta (almeno 65%), alcuni test svolti con perdite inferiori al 50% infatti dimostrano di non riuscire ad esprimere l'utilità della ritrasmissione, questo si spiega grazie al fatto che le ritrasmissioni standard del protocollo *Ieee802.11* sono sufficienti a sopperire alle interferenze ambientali "normali".
L'altro dato che si evince dalle tabelle è che la latenza sembra influenzare di poco il lavoro della ritrasmissione anticipata, si nota talvolta che il meccanismo lavori leggermente meglio con una latenza intermedia attorno ai 150ms.

Capitolo 9

Conclusioni

In questo elaborato lo scopo principale che ci si era prefissato era in primo luogo quello di mostrare quali sono le problematiche che affligono la tecnologia WiFi, soprattutto per quanto riguarda le interferenze ambientali; successivamente si voleva mostrare attraverso un framework di sviluppo simulativo una possibile soluzione ad una parte dei problemi (nella fattispecie si agisce sugli Access Point Wireless per incidere sulle comunicazioni TCP), che va ad integrare un'architettura per la QoS in continuo sviluppo e miglioramento.
Il lavoro svolto è partito innanzitutto con la ricerca e lo studio nel framework INET; questa fase è incentrata sullo scenario dell'implementazione preesistente, ABPS (soprattutto per quanto concerne i concetti), e soprattutto sui moduli componenti gli Access Point, oltre al livello MAC_RWMA modificato del nodo mobile. Questa è stata la parte più consistente del lavoro, necessaria per comprendere il funzionamento interno dei moduli e per partire con la progettazione vera e propria.
Successivamente si è passati allo studio delle possibili implementazioni delle modifiche da effettuare, si è quindi valutato quali scelte potevano essere le più semplici e concretamente sviluppabili. Per i motivi illustrati nel capitolo sulla progettazione, si è preferito procedere con l'implementazione di un sistema di ritrasmissione anticipata prioritaria, piuttosto che sul salvataggio del frame per il reinvio posticipato.
Tutta la fase seguente ha riguardato la scrittura del codice necessario accostandolo con il debugging ed il testing di pari passo alle singole modifiche; verranno tra poco spiegati alcuni problemi incontrati ed i possibili sviluppi.
Sono state raccolte, infine, le opportune misurazioni, seguite dalla loro analisi e da quelle delle problematiche incontrate, in base ai risultati ottenuti.

Problematiche Illustriamo qui alcune difficoltà incontrate nello sviluppo dell'elaborato.
Innanzitutto, nella prima fase di studio, si è affrontato il problema di dover adattare un'architettura simile ad ABPS, ad un Access Point Wireless, quindi ottenere la ritrasmissione anticipata senza però disporre di tutti i livelli di rete. Si ricorda infatti che INET fornisce AP con due soli livelli dello stack, fisico e datalink. Si è quindi optato per la soluzione illustrata.
In secondo luogo, ci si è scontrati con errori in fase di scrittura del codice e di testing, la cui difficile rilevazione e modifica è dovuta soprattutto alla natura strutturale ad albero della composizione dei moduli in OMNET++. Ci si imbatte quindi, ad esempio modificando un certo nodo di rete, in problematiche che vengono trasportate in livelli "lontani" dell'albero dei moduli, si è proceduto quindi con l'analisi dei risultati dei test e l'aiuto dell'ambiente simulativo grafico di OMNET.
Infine, in fase di esecuzione, si incontra estemporaneamente un errore riguardante la frammentazione dei pacchetti che blocca la simulazione in fase di testing, che appare nel modulo TCP del nodo mobile e viene notificato nel TCP modificato nelle preesistenti implementazioni di RWMA, riducendo la stabilità.

Sviluppi futuri Sono state individuate le seguenti possibili migliorie e sviluppi del sistema di ritrasmissione.

- Come prima cosa sarebbe utile, dopo uno studio approfondito di INET e del funzionamento dei moduli modificati nell'ambito RWMA, rendere stabile il meccanismo di ritrasmissione, individuando le cause specifiche delle situazioni di crash, probabilmente collegate con la frammentazione, ancora non implementata nel framework.

- Uno sviluppo interessante potrebbe essere l'implementazione descritta come scelta progettuale alternativa, quindi testare se l'agire con una ritrasmissione basata sul salvataggio del frame e il reinvio successivo, può eventualmente rivelarsi più efficiente.

- Come precedentemente accennato, in INET gli access point sono composti dai soli primi due livelli dello stack di rete. Una possibile implementazione futura potrebbe essere quella di comporre una nuova tipologia di AP, come un normale host, quindi sfruttare tutti i livelli dello stack per uniformare la ritrasmissione anticipata con quella preesistente di RWMA sul nodo mobile.

- Infine, sarebbe utile la seguente ottimizzazione del sistema. Come prima cosa si introdurrebbe l'individuazione della distanza fra client e server nelle connessioni TCP attive, al fine di monitorare ed applicare la ritrasmissione anticipata solo laddove è conveniente; questa feature attiverebbe il sistema RWMA solo qualora il Round Trip Time fosse abbastanza elevato da poter ridurre effettivamente la tempistica di ricezione dei pacchetti da parte del nodo mobile mediante la ritrasmissione locale.

Bibliografia

[1] Jonathan B. Spira (2003) - *20 YearsOne Standard: The Story of TCP/IP* - <http://www.cbi.umn.edu/iterations/spira.pdf>.

[2] Advanced TCP/IP - *The TCP/IP Protocols Details* - <http://www.tenouk.com/Module43.html>.

[3] E. Parton e H. de Groot - *Wireless a radiofrequenza per nuovi dispositivi medicali* - <http://www.elettronicanews.it/articoli/0,1254,40_ART_3627,00.html>.

[4] Michele Luti (2006) - *Ottimizzazioni del tcp su reti di accesso ieee802.11b/e e valutazioni delle prestazioni di Qos.*

[5] Selene Vincenzi - *Supporto al multihoming nei protocolli SIP e RTP: uno strumento simulativo.*

[6] Paolo Alberti - *ANALISI PER VIA SIMULATIVA DEL PROTOCOLLO TCP A RISTRASMISSIONE ASIMMETRICA ANTICIPATA SU WIFI.*

[7] Jacobson Van, Karels Michael (1988) - *Congestion Avoidance Control.*

[8] RFC 2001 - *TCP Slow Start, Congestion Avoidance, Fast Retransmit, and Fast Recovery Algorithms.*

[9] RFC 2582 - *The NewReno Modification to TCP's Fast Recovery Algorithm.*

[10] Sanadidi, Ren Wang, and Andrea Zanella, Claudio Casetti, Saverio Mascolo - *TCP Westwood: Congestion Window Control Using Bandwidth Estimation.*

[11] Lawrence S. Brakmo, Sean W. OMalley, Larry L. Peterson - *TCP Vegas: New Techniques for CongestionDetection and Avoidance.*

[12] INET/OMNET++ Documentation.

[13] Elisa Finotti (2005) - *Misure di perdita e ritardo di pacchetti in una rete wi-fi al fine di ottenere un modello del canale a livello di pacchetto - Disponibile a* <http://etd.adm.unipi.it/t/etd-06162005-122340>.

[14] Cheng Peng Fu, Associate Member, IEEE, and Soung C. Liew, Senior Member, IEEE (2003) - *TCP Veno: TCP Enhancement for Transmission Over Wireless Access Networks.*

[15] Orazio Mirabella - *Il protocollo 802.11.*

[16] IEEE Standards Association (2012)- IEEE 802.11 - *<http://standards.ieee.org/getieee802/download/802.11-2012.pdf>.*

[17] Giorgia Lodi Fabio Panzieri Antonio Messina Vittorio Ghini, Luigi Enrico Tomaselli - *Always best packet switching for sip-based mobile multimedia services.*

[18] IEEE 802.11 Frame - *http://www.technologyuk.net.*