UNIVERSITÀ DEGLI STUDI DI PISA

Facoltà di Scienze Matematiche, Fisiche e Naturali

Dipartimento di Informatica

Tesi Specialistica

Applicazione VoIP su reti veicolari IEEE 802.11p: misurazioni ed analisi delle prestazioni

Marco Volpetti

Relatore
Dott. Paolo Santi

Controrelatore
Prof. Augusto Ciuffoletti

Co-Relatore
Prof. Maurizio Bonuccelli

Anno Accademico 2010/2011

alla mia famiglia e a Nadia che mi sono sempre stati vicini

Indice

Elenco delle figure

Elenco delle tabelle

Capitolo 1

Introduzione

Una *rete veicolare ad-hoc* (*Vehicular Ad-Hoc Network o VANET*) è una tecnologia che sfrutta le auto in movimento come nodi di una rete per creare una rete mobile. Una rete di questo tipo trasforma ogni vettura partecipante in una sorta di router wireless o in un nodo. I veicoli che distano da 100 a 300 metri l'uno dall'altro possono essere sfruttati per il collegamento e, di conseguenza, per la creazione di una vasta rete. I veicoli i quali non rientrano all'interno della gamma del segnale o escono dalla rete non possono usufruire dei servizi che questa rete mette a disposizione.
Tra le *tecnologie wireless* emergenti usate dai conducenti e dai passeggeri, le comunicazioni wireless a corto raggio basate sullo *standard IEEE 802.11* hanno ricevuto una considerevole attenzione. Grazie ai bassi costi, alla disponibilità e all'ampia diffusione, ci si aspetta che i veicoli nel futuro saranno equipaggiati con unità a bordo comprendenti interfacce wireless ed antenne. La tecnologia wireless consente una completa e distribuita rete comunicativa tra i veicoli che viene comunemente chiamata *rete ad-hoc*. La combinazione della tecnologia wireless a corto raggio e delle reti *ad-hoc* porta alla nascita di un'ampia gamma di nuove applicazioni.
Nell'ambito molto ampio delle reti veicolari i laboratori NEC hanno progettato il sistema di comunicazione *CAR-2-X* il quale permette ai veicoli di comunicare tra loro e di comunicare con delle infrastrutture posizionate ai bordi della strada. Queste comunicazioni avvengono sfruttando la tecnologia wireless *IEEE 802.11p*, cioè una modifica dello standard *IEEE 802.11* originale effettuata appositamente per gli ambienti veicolari, e tramite le *LinkBird-MX*, cioè dei dispositivi hardware comprendenti software ed antenne wireless in grado di comunicare tra loro via etere. Il mondo delle reti veicolari è molto vasto e non comprende solo il sistema di comunicazione *C2X*, ma in questa tesi sarà l'unico sistema di comunicazione su reti veicolari che verrà preso in considerazione.

Questo sistema di comunicazione ha permesso lo sviluppo di nuove applicazioni riguardanti la sicurezza stradale, il miglioramento del traffico e l'intrattenimento di conducenti e passeggeri. Un esempio di quest'ultima tipologia di applicazioni potrebbe essere un'applicazione *VoiP (Voice over IP)* che, sfruttando l'architettura del sistema *CAR-2-X*, permetta ai conducenti e/o passeggeri di veicoli differenti di comunicare gratuitamente durante i loro viaggi.
Tutte queste applicazioni devono comunque confrontarsi con i vari problemi relativi alle tecnologie wireless come ad esempio la perdita dei pacchetti o il *jitter* (la variabilità dei tempi di interarrivo tra i pacchetti). Lo studio della comunicazione tra i veicoli e del comportamento di questi problemi al variare di certi parametri come ad esempio la distanza tra i veicoli, il *throughput* (la capacità di trasmissione del canale effettivamente utilizzata) ed altri è sicuramente un prerequisito fondamentale per la realizzazione di applicazioni che siano realmente utilizzabili in questi ambiti applicativi.

1.1 Descrizione del Lavoro

Questa tesi si colloca nell'ambito dei sistemi veicolari e delle tecnologie wireless e *VoIP*. Dopo un'iniziale studio di queste tematiche e dell'*SDK (Software development Kit)* fornito insieme alle *LinkBird-MX* è stato sviluppato un simulatore di pacchetti VoIP compatibile con l'hardware ed il software messo a disposizione.
Tale simulatore permette di impostare e modificare molti parametri utili per diversificare il tipo e la quantità del traffico. Inoltre sia il dispositivo che invia sia il dispositivo che riceve genereranno un file di log contenente i dati dell'esperimento.
Un'ulteriore programma di post-processing è stato realizzato al fine di poter analizzare i file di log generati durante l'esperimento per calcolare parametri come il jitter, il throughput e la distanza tra i veicoli in funzione del tempo. Il programma di post-processing genera così altri due file di log: uno molto dettagliato contenente tutti i dati del post-processing ed uno molto schematico da poter utilizzare con un programma esterno per generare automaticamente grafici che mostrino i risultati ottenuti.
Sia il simulatore sia il programma di post-processing sono stati integrati in una semplice ed intuitiva interfaccia grafica. Fatto ciò siamo passati agli esperimenti che hanno avuto luogo in prima istanza nei laboratori riproducendo una situazione ottimale per poi passare in alcune zone della città di Pisa per sperimentare situazioni di traffico reali. Infine abbiamo elaborato i dati ricavati e confrontato i risultati.
Questa tesi è stata svolta presso il *Consiglio Nazionale delle Ricerche (C.N.R.)*

di Pisa sotto la supervisione del *Dott. Paolo Santi* ed ha avuto una durata complessiva di circa 9 mesi.

1.2 Obiettivi della Tesi

Possiamo dividere gli obiettivi di questa tesi in due parti, un'obiettivo principale su cui si è fondata totalmente la tesi, ed altri obiettivi secondari, ma non per questo meno importanti o meno difficili da raggiungere, che sono diretta conseguenza dei risultati raggiunti e dell'evoluzione della tesi. L'obiettivo principale è quello di riuscire a capire se è possibile far girare correttamente un'applicazione *VoIP* per i veicoli facenti parte di un ambiente veicolari e comunicanti tramite interfacce wireless. Far girare correttamente un'applicazione *VoIP* significa verificare che ci siano le condizioni adatte per far sì che due veicoli, entrambi sfruttando l'eventuale applicazione *VoIP*, riescano a comunicare correttamente senza che la qualità della voce non ne risenta troppo o, ancor peggio, che sia del tutto impossibilitata.
L'avanzamento degli esperimenti svolti e l'analisi dei risultati ci ha portato a nuovi obiettivi sotto forma di interrogativi a cui volevamo dare risposta. Per prima cosa abbiamo analizzato il canale wireless utilizzato e ci siamo chiesti se tale canale avesse un comportamento simmetrico o asimmetrico. Inoltre, nella fase finale degli esperimenti, ci siamo chiesti se un'eventuale applicazione di beaconig (applicazioni che sono indispensabili negli ambienti veicolari) influenzasse i risultati o meno. Infine ci siamo chiesti se la variazione della frequenza di invio incidesse sui risultati o meno. Tutti gli obiettivi che ci siamo prefissati sono stati raggiunti pienamente.

1.3 Lo Stato dell'Arte

Per quanto riguarda lo stato dell'arte delle tecnologie che stanno alla base di questa tesi possiamo dire che le reti veicolari sono una tecnologia in costante evoluzione che necessita ancora di test e tempo per un'ampia diffusione sul mercato. La tecnologia wireless che sfrutta lo standard *IEEE 802.11* di contro è ormai una tecnologia affermata e rappresenta lo standard per le connessioni wireless. Lo standard *IEEE 802.11* è stato modificato in molti modi per essere adattato ai diversi ambiti cui le connessioni wireless fanno parte, ed in particolare lo standard *IEEE 802.11p*, usato negli ambienti veicolari, è diventato da poco (luglio 2011) ufficialmente standard *IEEE*.
Il *VoIP*, come lo standard *IEEE 802.11*, risulta ormai una tecnologia consolidata ed ampiamente utilizzata per le conversazioni telefoniche che sfruttano le

connessioni internet, basti pensare al grado di diffusione di programmi che sfruttano questa tecnologia come *Skype* [25]. Quello che non è stato ancora studiato, e che è proprio quello di cui tratta questa tesi, è lo studio dell'applicazione della tecnologia *VoIP* su reti veicolari sfruttando comunicazioni wireless.
Per quanto riguarda le misurazioni su campo di reti veicolari *IEEE 802.11p* possiamo citare i seguenti articoli. Nell'articolo [26], dopo un'iniziale trattazione sulle differenze tra lo standard *IEEE 802.11p* e gli standard precedenti della famiglia *IEEE 802.11*, viene spiegato come effettuare misurazioni su ambienti veicolari comunicanti tramite interfacce wireless *IEEE 802.11p* con le apparecchiature di prova *Rohde & Schwarz (R&S)* compatibili con la famiglia degli standard *IEEE 802.11*. L'articolo [27] si concentra sulla comparazione tra i parametri del canale *IEEE 802.11p* con i parametri valutati a seguito di nuove misurazioni ad alta velocità effettuate a Lund in Svezia e in una autostrada. Invece nell'articolo [28] vengono valutate le performance a seguito di misurazioni in uno scenario *infrastructure-to-vehicle (I2V)* all'interno di un tunnel nell'autostrada S1 vicino Vienna. Nell'articolo [29] viene presentata una simulazione realistica di un modello per *IEEE 802.11p* in ambiente urbano. In questo articolo, tramite l'uso di dispositivi per le misurazioni che sfruttano lo standard *IEEE 802.11p*, vengono stimati gli effetti che hanno costruzioni e altri ostacoli tra i veicoli sulle comunicazioni wireless *IEEE 802.11p*. Infine nell'articolo [24] sono state effettuate misurazioni sulle reti veicolari basate su *IEEE 802.11p* concentrandosi sullo scambio di messaggi di beaconing tra veicoli contenenti informazioni di stato per le applicazioni di sicurezza. Le misurazioni di quest'ultimo articolo sono state svolte sullo stesso tratto stradale su cui si è svolta questa tesi e con lo stesso hardware.
Per quanto riguarda invece i lavori riguardanti misurazioni *VoIP* in ambienti wireless possiamo citare i seguenti articoli. L'articolo [30] si occupa di analizzare i risultati ottenuti a seguito di misurazioni sul traffico *VoIP* in ambienti wireless focalizzandosi sulla capacità massima del traffico prendendo soprattutto in considerazione il tempo di pacchettizzazione ed il numero di chiamate effettuate. Invece nell'articolo [31] si prende in considerazione il cambio di veloctià trasmissiva durante una sessione *VoIP* che porta ad una degradazione del servizio. Viene proposta dunque una nuova versione centralizzata di codec con algoritmi adattivi che permettono ottimizzazioni delle risorse di rete e permettono di migliorare la qualità della voce. Nell'articolo [7] viene preso in considerazione il traffico vocale su comunicazioni wireless basate sullo standard *IEEE 802.11e* e vengono analizzate le misurazioni effettuate a seguito della modifica di alcuni parametri di questo standard. Concludiamo citando [32] che si occupa dettagliatamente di analizzare la qualità del servizio del traffico *VoIP* su reti wireless *IEEE 802.11*.

1.4 Organizzazione della Tesi

Questa tesi è strutturata nel seguente modo.
Il Capitolo 2 riassume e descrive le principali tecnologie utilizzate in questa tesi.
Il Capitolo 3 descrive il setup fisico necessario per svolgere gli esperimenti, le tecnologie e i dispositivi utilizzati, il software utilizzato e i parametri presi in considerazione.
Il Capitolo 4 descrive dettagliatamente tutti gli esperimenti svolti e i risultati ottenuti.
Infine nel Capitolo 5 vengono trattate le conclusioni della tesi con particolare riguardo alle esperienze acquisite e alle eventuali estensioni che potrebbero essere apportate.

Capitolo 2

Le Tecnologie Utilizzate

Questo capitolo introduce e descrive le reti veicolari, la tecnologia wireless *IEEE 802.11p* e il *VoIP* su reti veicolari, cioè le principali tecnologie che sono alla base di questa tesi. Poiché questa tesi ha proprio come ambito le reti veicolari e la tecnologia *VoIP* applicata ad esse sfruttando principalmente la tecnologia wireless *IEEE 802.11p* per lo scambio di informazioni, una conoscenza, perlomeno basilare, di queste tematiche risulta fondamentale per una corretta comprensione dei risultati ottenuti e delle scelte progettuali fatte.
Vedremo innanzitutto una descrizione generica delle reti veicolari, il loro scopo e le loro componenti fondamentali soffermandoci principalmente sulle parti più importanti per i nostri scopi ma senza addentrarci troppo in dettagli per noi inutili. Fatto ciò analizzeremo brevemente l'evoluzione tecnologica che ha portato alla modifica dello standard originale *IEEE 802.11* a favore di *IEEE 802.11p* per l'uso in ambito veicolare. Anche in questo caso tralasceremo inutili descrizioni tecniche degli standard per concentrarci invece sulle fondamentali differenze che hanno permesso allo standard *IEEE 802.11p* di adattarsi perfettamente agli ambienti veicolari. Questa evoluzione tecnologia ci porterà anche ad una breve descrizione dello standard *IEEE 802.11e*. Concluderemo poi descrivendo gli aspetti più importanti del *VoIP* su ambienti veicolari soffermandoci principalmente sui tipici problemi di questa tecnologia.

2.1 Le Reti Veicolari

Cominciamo dunque col presentare le reti veicolari in generale ed il sistema di comunicazione che esse usano. Possiamo definire, ad alto livello, una rete veicolare come un'insieme di unità eterogenee, ognuna delle quali facente parte di un dominio ben specifico che comunicano tra loro tipicamente tramite interfacce wireless a corto raggio con lo standard *IEEE 802.11p* oppure tramite altri tipi

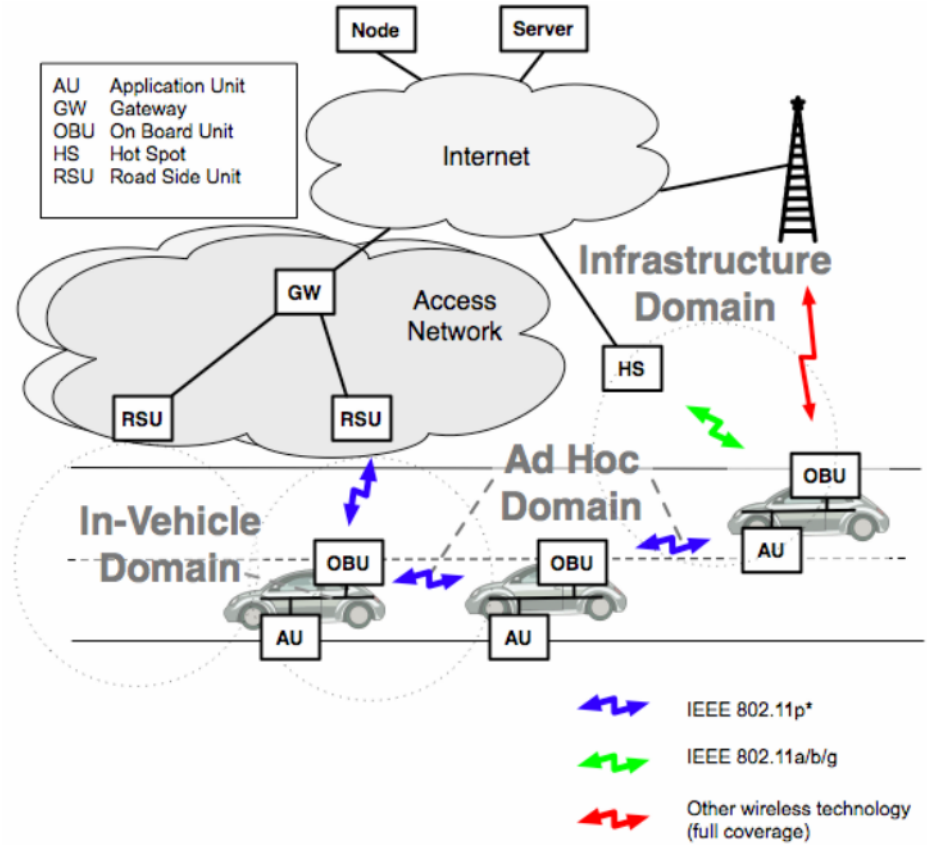

Figura 2.1: Architettura di una rete veicolare in cui vengono mostrati i diversi domini che ne fanno parte, le varie unità di cui ogni dominio è composto e le comunicazioni usate [3]

di connessione a seconda del dominio di appartenenza.
Quello che caratterizza le reti veicolari e le differenzia dalle classiche reti internet è che sono state progettate per l'ambiente stradale e che quindi comprendono i veicoli che viaggiano su strada e devono fare i conti con una serie di problemi che nascono a causa della natura caotica del traffico. Gli obiettivi primari delle reti veicolari sono l'aumento della sicurezza stradale, il miglioramento del gestione del traffico e la fornitura di applicazioni di intrattenimento per i conducenti dei veicoli e per i loro passeggeri. Questi obiettivi vengono raggiunti tramite l'uso di applicazioni di diverso tipo che sfruttano via via diverse componenti di queste reti.

2.1.1 L'architettura di una rete veicolare

La figura 2.1 ci mostra come può essere struttura una rete veicolare. Possiamo innanzitutto individuare tre macro-domini ognuno dei quali contenente varie unità che svolgono compiti diversi. Il dominio all'interno del veicolo (*In-Vehicle Domain*) è un dominio molto piccolo che, come si intuisce dal nome, caratterizza un singolo veicolo e che è formato da un'unità a bordo (*On-Board Unit* o *OBU*) e da una o più unità applicative (*Application Unit* o *AU*).
Le *AU* sono unità che risiedono unicamente all'interno dei veicoli e che svolgono un particolare compito sfruttando la *OBU* a cui sono collegate per comunicare all'esterno del dominio. Le *AU* sono le unità su cui andranno a girare le applicazioni che andranno ad interagire con i conducenti e con i passeggeri dei veicoli. La *OBU* è tipicamente un dispositivo dotato di antenna che comunica tramite interfaccia wireless *IEEE 802.11p* all'esterno del dominio, ma può essere dotata anche di altri dispositivi di rete che sfruttano altre tecnologie radio. Ad ogni *OBU* possono essere collegate una o più *AU* tramite connessione cablata o di altro tipo (wireless, Bluetooth o altre), inoltre queste due tipi di unità possono risiedere anche in un unico componente fisico.
Quindi in una rete veicolare sono previsti molti domini all'interno del veicolo che comunicano tra loro tramite le *OBU* sfruttando la tecnologia *IEEE 802.11p*. Queste comunicazioni rientrano nel dominio *ad-hoc* che comprende tutti quei veicoli che hanno una *OBU* (quindi comprende i vari domini all'interno del veicolo) e le unità fisse che risiedono ai bordi delle strade che vengono denominate unità a bordo strada (*Road-Side Unit* o *RSU*). Possiamo immaginarci una *RSU* come un dispositivo fisico (tipicamente un sensore) che comunica tramite un'interfaccia di rete wireless *IEEE 802.11p* con le varie *OBU*. Le *RSU* possono comunicare anche al di fuori del dominio *ad-hoc* per fornire connettività Internet alle altre unità ad esse connesse, e possono anche essere equipaggiate con altri dispositivi di rete.
Grazie alle *RSU* le *OBU* possono comunicare tra loro anche se non sono direttamente collegate sfruttando comunicazioni *multi-hop*. In realtà anche senza le *RSU* è possibile offrire comunicazioni *multi-hop* tramite le *OBU* che inoltrano il traffico dati tra di loro senza bisogno di intermediari. Infine abbiamo anche un dominio delle infrastrutture (*Infrastructure Domain*) che comprende le varie *RSU* e tutti quei dispositivi come *Hot Spot (HS)* o *Gateway (GW)* che permettono l'accesso ai tradizionali servizi internet. In questo modo una *AU* può accedere direttamente ai servizi internet sfruttando la *OBU* a cui è collegata che a sua volta sfrutterà una o più *RSU* per accedere al dominio delle infrastrutture.
Nell'ambito di questa tesi non andremo a sfruttare tutti i possibili domini e tutte le possibili unità delle reti veicolari. Infatti nel nostro scenario abbiamo due

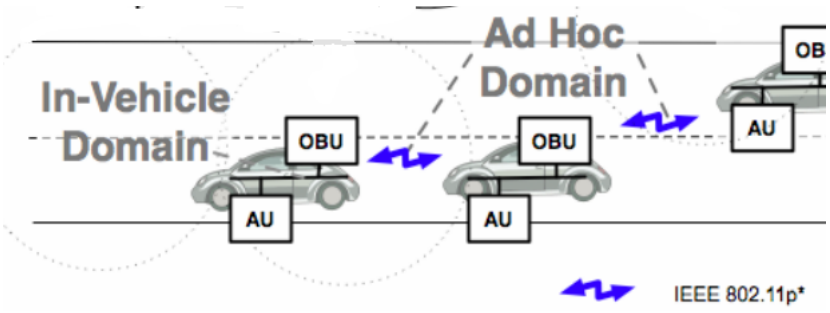

Figura 2.2: Architettura di una rete veicolare ristretta ad alcune unità.

domini all'interno del veicolo ognuno dei quali composto da una *OBU* e da una *AU* che gira su un portatile collegato tramite connessione cablata alla *OBU*. Le due *OBU* dei due veicoli si trasmettono informazioni direttamente senza connessioni multi-hop, di conseguenza non sono previste *RSU* o altre unità. Ignoriamo dunque il dominio delle infrastrutture e semplifichiamo al massimo il dominio *ad-hoc* sfruttando comunicazioni dirette. La figura 2.2 ci mostra l'architettura di una rete veicolare ristretta alle unità che ci interessano.

2.1.2 Il sistema di comunicazione di una rete veicolare

I laboratori NEC hanno progettato il *sistema di comunicazione CAR-2-X* per le reti veicolari. Il termine *CAR-2-X* (*C2X*) va interpretato con due possibili significati, il *CAR-2-CAR* (*C2C*) che indica il sistema di comunicazione che permette le interazioni dirette tra i veicoli (tra le *OBU* direttamente) e il *CAR-2-Infrastructure* (*C2I*) che indica il sistema di comunicazione che permette le interazioni tra i veicoli e le infrastrutture (tra le *OBU* e le *RSU*).
Questo sistema di comunicazione si basa sulle comunicazioni wireless a corto raggio e cerca di fornire un'ampia diffusione spaziale delle informazioni e alla velocità maggiore possibile in modo da poter essere sfruttata soprattutto per le applicazioni di sicurezza che richiedono tempi di reazione molto bassi. Inoltre, con questo sistema di comunicazione, non è necessaria un'infrastruttura di rete pre-installata. L'architettura di questo sistema di comunicazione mette a disposizione vari livelli e protocolli:

- il *livello delle applicazioni* che fornisce servizi applicativi grazie ai quali si possono creare applicazioni che interagiscano con conducenti e passeggeri,
- il *livello di rete* che fornisce le funzionalità per lo scambio delle informazioni all'interno delle reti veicolari tramite appositi algoritmi di routing e con diversi schemi di trasmissione come il *broadcast geografico* o il *broadcast single-hop*,

- il *livello MAC* che si basa sul protocollo *MAC* di *IEEE 802.11* sfruttando lo standard *Carrier Sense Multiple Access con Collision Avoidance* (*CSMA/CA*) ma con semplificazioni e miglioramenti,
- il *livello fisico* che sfrutta la tecnologia *IEEE 802.11p* che deriva direttamente da *802.11a* ma con apposite modifiche per l'ambiente veicolare.

Dobbiamo sottolineare che in questa tesi, benché si usino i *livelli fisico* e *MAC* di questo sistema di comunicazione, si sfruttano il protocollo *UDP* per quanto riguarda il *livello di trasporto* e lo standard *IPv4* per quanto riguarda il *livello di rete* anziché i nuovi protocolli introdotti dal sistema di comunicazione *C2X*. Va altresì sottolineato che affermando che sfruttiamo il protocollo di livello di rete *IPv4*, non intendiamo dire che sfruttiamo o lavoriamo sulla rete Internet, ma che sfruttiamo le tecnologie standard *IPv4* applicate alle reti veicolari. Va infine notato che la scelta di *IPv4* è anche forzata dal fatto che vogliamo valutare le prestazioni del *VoIP*.

2.2 Le Tecnologie Wireless

Tutte le comunicazioni e lo scambio di informazioni che sono state sfruttate in questa tesi si sono basate sullo standard wireless *IEEE 802.11p*. Daremo quindi una descrizione di questo standard soffermandoci principalmente sugli aspetti che più ci interessano di questa tecnologia e sui miglioramenti apportati rispetto ai precedenti standard per l'adattamento all'ambiente veicolare. Prima di questa trattazione vedremo però una breve descrizione della tecnologia wireless 802.11 in generale e di alcuni standard precedenti per una maggiore chiarezza.
Quello in generale delle reti wireless è un ambito molto ampio e complesso pieno di tecnologie, definizioni e componenti hardware e software. Cercheremo qua di tralasciare tutti quegli aspetti che non sono fondamentali per i nostri scopi e che andrebbero solamente ad appesantire la lettura. Ci concentreremo invece inizialmente sullo standard *IEEE 802.11* per poi descrivere la sua evoluzione rappresentata dallo standard *IEEE 802.11e* che si avvicina ai nostri scopi per poi concludere descrivendo il nuovo standard *IEEE 802.11p* usato negli ambienti veicolari.

2.2.1 Gli Standard IEEE 802.11 ed IEEE 802.11e

Cominciamo col descrivere lo standard *IEEE 802.11*, creato dal gruppo di ricerca *11* dell'*IEEE 802*, che lavora sul *livello fisico (PHY)* e sul *livello MAC* del modello *ISO/OSI* e sta alla base di tutti quei dispositivi che forniscono connettività nelle reti locali wireless (*WLAN*) usati nel *Wi-Fi Alliance* (*Wi-Fi*). Avere

una corretta e completa conoscenza di questo standard richiede molto tempo ed impegno, vedremo qua solo una descrizione generica di questo standard con i suoi componenti fondamentali ed una breve descrizione dei meccanismi di contesa che utilizza.
Cominciamo col definire una componente fondamentale e cioè l'infrastruttura *Basic Service Set* (*BSS*) che è formata da un gruppo di stazioni radio che sfruttano lo standard *IEEE 802.11* e che sono ancorate ad un *Access Point* (*AP*) e configurate in modo tale che le stazioni radio possano comunicare tra loro tramite il collegamento wireless. All'interno di una *BSS* ci sono appositi meccanismi grazie ai quali è possibile controllare l'accesso alle risorse e ai servizi di un *AP*. Inoltre, all'interno di una *BSS*, le sorgenti radio possono filtrare le trasmissioni radio provenienti da altre sorgenti estranee nelle vicinanze. Il meccanismo che permette ad una sorgente radio di unirsi ad una *BSS* prevede che tale sorgente radio per prima cosa aspetti un messaggio di segnalazione proveniente da un *AP* e poi si unisca alla *BSS* tramite una serie di passaggi iterativi che comprendono l'autenticazione e l'associazione. Questo è uno dei motivi per cui lo standard *IEEE 802.11* non è adatto all'ambiente veicolare infatti queste procedure richiedono troppo tempo. Lo standard *IEEE 802.11* consente inoltre di combinare logicamente insieme uno o più *BSS* interconnessi in un *Extended Service Set* (*ESS*), utilizzando il *Distribution Service* (*DS*). Inoltre è possibile avere anche un'infrastruttura particolare chiamata *Independent BSS* (*IBSS*) che però risulta ancora troppo complessa e dispendiosa per essere adattata per le comunicazioni veicolare.
Un'altra componente fondamentale è il *Service Set Identification* (*SSID*) grazie al quale una *BSS* risulta nota e raggiungibile agli utenti. Va notato infine che l'*SSID* non deve essere confuso con il *BSSID*, che sta per *Basic Service Set Identification*. Diversamente dall'*SSID*, il *BSSID* è il nome di un *BSS* noto alle sorgenti radio a livello *MAC* ed è un campo lungo 48 bit, proprio come un indirizzo MAC. Ogni *BSS* deve avere un unico *BSSID* condiviso da tutti i membri e ciò viene semplicemente garantito utilizzando l'indirizzo *MAC* dell'*AP*. Il livello *MAC* di *IEEE 802.11* si occupa quindi di come organizzare una serie di sorgenti radio, al fine di stabilire e mantenere un gruppo cooperante, ed inoltre determina quando un pacchetto deve essere inviato e definisce le regole di base per farlo. Invece il livello fisico trasforma i bit in qualunque mezzo fisico si intenda utilizzare per trasmettere i dati.
Come abbiamo già accennato questo standard, a *livello MAC*, utilizza dei meccanismi di contesa per accedere al mezzo. Dal momento che nelle reti wireless *IEEE 802.11* i nodi non possono sia inviare sia ascoltare il mezzo contemporaneamente, e di conseguenza non possono accorgersi di eventuali collisioni, si utilizza lo schema *Carrier Sense Multiple Access* con *Collision Avoidance*

(*CSMA/CA*) per evitare che avvengano conflitti dovuti ad invii multipli contemporanei nel mezzo. La descrizione di questi meccanismi non è necessaria, basti sapere che tipicamente viene utilizzato il meccanismo di contesa *DCF* (*Distributed Coordination Function*) in cui tutti i parametri che esso utilizza sono fissati e non modificabili.
Prima di arrivare a descrivere lo standard *IEEE 802.11p* menzioniamo anche lo standard *IEEE 802.11e* il quale, modificando l'originale standard *IEEE 802.11*, prova a fornire garanzie sulla qualità del servizio (*QoS*) tramite cambiamenti a livello *MAC*. Questo standard permette la modifica dei parametri a livello *MAC* e permette di aggiungerne altri allo scopo di alterare i meccanismi di contesa mentre nel precedente standard questi parametri erano fissati. Più precisamente le code dei pacchetti da trasmettere vengono differenziate a seconda dell'appartenenza a definite *Access Category*. Queste sono caratterizzate da tempi di accesso al servizio differenti, in modo da garantire differenti livelli di priorità. Normalmente i servizi che richiedono minori ritardi di trasmissione (come il *VoIP*) sono associati a livelli di priorità più elevati. Al contrario il traffico *Best Effort* viene associato ad una priorità inferiore.

2.2.2 Lo Standard IEEE 802.11p

Veniamo finalmente allo standard *IEEE 802.11p*, utilizzato in questa tesi, di cui siamo principalmente interessati a capire i miglioramenti rispetto ai precedenti standard grazie ai quali è stato possibile sfruttarlo negli ambienti veicolari. Le modifiche allo standard originale sono state necessarie per far fronte ai requisiti stringenti delle reti veicolari che impongono tempi di connessione molto brevi sia per le applicazioni di sicurezza sia per le altre applicazioni. A questo nuovo standard, conosciuto anche come *IEEE 802.11p WAVE* (*Wireless access in vehicular envoriments*), sono state allocate delle bande di frequenza sia negli Stati Uniti sia in Europa che sono state denominate *Dedicated short range Communications* (*DSRC*). Il *DSRC* è strutturato in sette canali dall'ampiezza di 10 Mhz. Il canale 178 è quello di controllo (*CCH*) ed è ristretto alle comunicazioni di sicurezza. I canali 184 e 172 sono riservati per usi speciali mentre gli altri canali sono i canali di servizio (*SCH*) che vengono usati sia per le applicazioni di sicurezza sia per quelle non di sicurezza. In questa tesi sfrutteremo sia i canali di servizio sia il canale di controllo.
I miglioramenti a livello *MAC* di questo nuovo standard sono orientati ad abbattere i tempi di connessione troppo lunghi del precedente standard. Infatti necessitiamo di uno scambio dati pressoché istantaneo e non ci possiamo permettere la scansione dei canali per i messaggi di segnalazione di un *BSS* ed eseguire le procedure di handshakes per stabilire la comunicazione. Risulta quindi

fondamentale per le sorgenti radio *IEEE 802.11p* essere nello stesso canale ed essere configurate con lo stesso *BSSID*. Più precisamente possiamo riassumere i miglioramenti a livello *MAC* di questo nuovo standard con i seguenti:

- la *modalità WAVE* la quale prevede che le stazioni radio che sono in questa modalità possano trasmettere e ricevere frame di dati con il carattere speciale *BSSID* senza far parte a priori di una *BSS*,
- l'infrastruttura *WBSS* (*BSS WAVE*) tramite la quale una stazione radio può decidere di unirsi ad una *WBSS* solo ricevendo un annuncio *WAVE* abbassando notevolmente in questo modo l'overhead dovuto ai tempi di collegamento,
- l'utilizzo del carattere speciale *BSSID* usato per raggiungere tutte le stazioni radio limitrofe.

Per quanto riguarda invece i miglioramenti a livello fisico dobbiamo tener conto che sono state fatte le minime modifiche possibili onde evitare il cambio dell'intera tecnologia per i collegamenti wireless. Più precisamente possiamo riassumere i miglioramenti a livello *PHY* di questo nuovo standard con i seguenti:

- l'uso dei canali a *10 Mhz* con l'*Orthogonal Frequency-Division Multiplexing* (*OFDM*) anziché l'uso di canali a *20 Mhz* per affrontare il maggiore ritardo *RMS (Root-Mean-Square)* presente negli ambienti veicolari,
- il miglioramento delle performance del ricevitore tramite respingimenti nei canali adiacenti per evitare le interferenze tra veicoli vicini che operano su canali adiacenti,
- il miglioramento delle maschere di trasmissione tramite requisiti più stringenti.

2.3 Il VoIP su ambienti veicolari

Descriviamo ora brevemente il *VoIP* (*Voice over IP*) in ambito veicolare anziché sulla base delle tradizionali reti internet. Poiché l'obiettivo primario di questa tesi è quello di capire se è possibile realizzare un'applicazione *VoIP* da sfruttare per gli ambienti veicolari, è necessario descrivere questa tecnologia, le sue caratteristiche principali e i problemi da cui è affetta. Molte di queste caratteristiche e problemi, prese dallo studio nelle classiche reti internet, sono presenti anche nelle reti veicolari.

Il *VoIP* risulta ormai una tecnologia molto conosciuta e diffusa nelle reti tradizionali, ma non sono stati fatti studi o lavori per quanto riguarda le reti veicolari. In generale lo scopo del *VoIP* è quello di portare la voce tramite una rete sfruttando tecnologie già diffuse ed esistenti. Nel nostro caso con l'uso del VoIP intendiamo la possibilità di portare la voce sfruttando le reti veicolati, quindi sarebbe più opportuno definirlo *Voice over Vehicular Networks* (*VoVN*). Nonostante l'ambito applicativo sia diverso da quello tradizionale, dobbiamo fare i conti con gli stessi problemi quindi andiamo ora a descrivere alcune delle caratteristiche e delle problematiche più comuni del *VoIP*.

2.3.1 Il Ritardo

Un primo parametro di cui dobbiamo tener conto è il ritardo (o latenza) che possiamo definire come il tempo che impiega la voce del parlante a raggiungere l'orecchio del destinatario ed esistono tre tipi di ritardi, variabili in funzione del canale fisico e di altri fattori:

- il ritardo di propagazione che è dovuto alla propagazione del segnale dal mittente al destinatario. Con canali a basso bit-rate è sicuramente trascurabile mentre diventa il ritardo preponderante nei collegamenti a lunga distanza e quando le altre cause di ritardo risultano limitate,
- il ritardo di elaborazione che è dovuto ai processi di elaborazione numerica dei segnali, alla compressione e commutazione dei pacchetti,
- il ritardo di accodamento che è dovuto, qualora si verifichi una congestione, ai pacchetti che devono essere inseriti in un buffer e poi gestiti da una coda.

2.3.2 Il Jitter

Un problema molto serio della trasmissione in tempo reale è la sincronizzazione e la continuità del flusso dati. Se i pacchetti vengono persi o arrivano con dei ritardi assai diversi fra loro, il ricevitore fatica ad estrapolare da queste informazioni un flusso regolare di dati. Il concetto di *jitter* può essere associato alla variabilità dei tempi di interarrivo tra i pacchetti (che sono in realtà processi aleatori). Possiamo definire semplicemente *jitter* come la differenza fra il momento in cui il pacchetto è atteso e il momento in cui viene ricevuto.
Per contrastare questo effetto si utilizzano dei buffer di accodamento (e degli algoritmi opportuni) che regolarizzano il flusso in uscita. Ovviamente più il *jitter* è elevato, e più questi buffer devono essere capienti. Di contro, buffer molto grandi introducono ulteriori ritardi (di accodamento). La soluzione ottima sta allora nell'adozione di buffer dinamici che possano cambiare dimensione in funzione dello stato della rete.

2.3.3 Le Codifiche Vocali

Le codifiche vocali vengono utilizzate per ridurre il bit-rate, così da poter trasmettere in condizioni meno favorevoli di banda e canale. Questi codici divengono indispensabili quando si trattano segnali multimediali come appunto la voce. Una delle principali funzioni delle codifiche vocali (o codec audio), è quella di limitare il bit-rate dei dati utili per garantire la banda per alcuni bit di controllo d'errore. Il ricevitore decodifica i campioni originali e dunque ricostruisce il segnale. Esistono tre tipi di codifiche:

- *Waveform coders* (codifica di forma d'onda): cercano di riprodurre nel modo più fedele possibile la forma d'onda, incluso il rumore di sottofondo. Producono dei campioni di alta qualità, ma ad alto bit-rate (la stessa codifica *ITU-T G.711* ossia il *PCM*, è di questo tipo),
- *Vocoders* (voice coders): non riproducono la forma d'onda, ma estraggono dei parametri dal campione e li inviano al ricevitore che da essi sintetizza il campione attraverso un filtro digitale a coefficienti variabili,
- *Hybrid coders*: uniscono i vantaggi dei due precedenti, operando a bit-rate molto bassi (4-16 kbit/s).

In questa tesi è stata usata quasi totalmente la codifica vocale *G.711*, basata sulla codifica *PCM*, nella quale il segnale audio viene campionato con una frequenza di 8 kHz, ottenendo pertanto 8000 campioni al secondo. Ogni campione viene quantizzato a 256 livelli e rappresentato con 8 bit, quindi il flusso numerico risultante è di 64 kbit/s.

2.3.4 La Qualità del Servizio

Con il termine Qualità del Servizio (*QoS*) si intendono i tentativi di modellare il traffico in modo da garantire certi parametri in termini di ritardo e di distribuzione del traffico. Dobbiamo tener conto che la perdita di pacchetti diminuisce la qualità della voce e code di grandi dimensioni limitano la perdita di pacchetti, però pacchetti e code di grandi dimensioni aumentano il ritardo, peggiorando la qualità della voce.
Se prendiamo ad esempio la codifica vocale *G.711*, utilizzata in questa tesi, si ha che una perdita di 32-64 ms risulta distruttiva, in quanto porta alla mancanza di fonemi, mentre buchi di 4-16 ms non sono percepibili dall'ascoltatore. La dimensione delle code è una questione molto delicata perché si tratta di scegliere un compromesso tra ritardo e probabilità di perdita di pacchetti.

2.3.5 La Perdita di pacchetti

La perdita di pacchetti è uno dei problemi più importanti ed esistono diverse soluzioni che mirano a porvi parziale rimedio. L'obiettivo principale in questo caso non è ricevere il messaggio integro e corretto, ma riuscire a ricostruire il parlato con sufficiente fedeltà da soddisfare l'ascoltatore più attento. Pertanto la priorità assoluta diventa minimizzare il ritardo e il *jitter*. Una possibile soluzione a questo problema consiste nel ripetere l'ultimo campione (strategie di *Packet Loss Concealment*), ma esistono anche altre strategie più complesse che interpolano il campione mancante sfruttando la forte correlazione fra i campioni vocali adiacenti.

2.3.6 Altri aspetti relativi al VoIP

Il *VoIP* risulta essere un argomento molto vasto e complesso ed una sua completa trattazione esula dagli scopi di questa tesi. Oltre a ciò che è stato descritto sopra, va ricordato che nel *VoIP* l'eco fa parte di ogni normale conversazione telefonica, perché quando si parla la voce giunge anche attraverso la scatola cranica sfasata di qualche millisecondo. Ascoltare la propria voce con un ritardo superiore a 25 ms diventa fastidioso e quindi questo risulta un problema che va preso in considerazione.
Inoltre dobbiamo tenere presente che la voce è un segnale analogico che varia lentamente nel tempo, in cui si alternano periodi di parlato e di silenzio. Questi periodi possono essere approssimati da una distribuzione esponenziale con media di 352 ms per il parlato e 650 ms per il silenzio (come mostrato nell'articolo [16]). Inoltre va notato che questi valori sono indicativi dal momento che in altri articoli variano ([14],[15],[17], [18]). Noi abbiamo preso come valori medi un secondo come media del parlato ed un secondo e mezzo come media del periodo di silenzio.
Infine va notato che per ridurre ulteriormente il bit-rate si utilizza una tecnica chiamata *VAD (Voice Activity Detection*) che riconosce i periodi di solo silenzio, analizzando principalmente il livello del segnale utile e del rumore di sottofondo. In questi periodi di solo silenzio si evita di trasmettere e verranno poi riempiti con del rumore artificiale (*comfort noise*), per combattere la sgradevole sensazione di mancanza di segnale.

Capitolo 3

I dettagli degli esperimenti

Questo capitolo descrive dettagliatamente le tecnologie hardware e software utilizzate per gli esperimenti svolti in questa tesi, con particolare attenzione agli aspetti più rilevanti dell'implementazione. Tutto il codice sviluppato, sia per quanto riguarda la parte del Simulatore e del programma di Post-Processing, sia per quanto riguarda la parte dell'interfaccia utente, è stato scritto nel linguaggio di programmazione *Java* usando la versione *1.6 update 20* delle librerie, utilizzando l'ambiente di sviluppo *Eclipse*, ed il tutto sotto il sistema opertivo *Linux* con il *kernel 2.6.35*.
Gli esperimenti svolti possono essere suddivisi in due categorie:

- esperimenti in laboratorio: questi esperimenti sono stati svolti nei laboratori del *Consiglio Nazionale delle Ricerche (C.N.R.) di Pisa* allo scopo di testare il sistema in una situazione ottimale,
- esperimenti su strada: questi esperimenti sono stati svolti guidando lungo un tratto stradale della città di *Pisa* allo scopo di testare il sistema in situazioni reali di traffico.

3.1 Il Setup Fisico

Poiché in questa tesi sono state usate due unità *LinkBird-MX*, cominciamo descrivendo queste unità hardware per poi continuare descrivendo il setup fisico necessario per gli esperimenti in laboratorio per poi presentare le componenti aggiuntive, rispetto al caso in laboratorio, necessarie per gli esperimenti su strada.

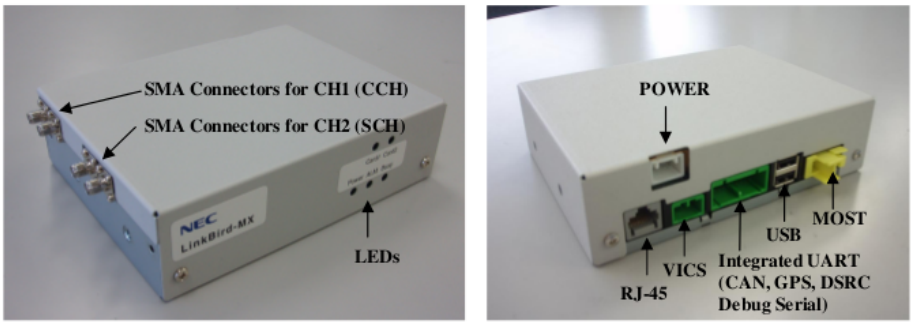

Figura 3.1: Due immagini di una LinkBird-MX, una frontale l'altra dal retro [1]

3.1.1 Le LinkBird-MX

Le *LinkBird-MX*, di cui possiamo vedere un esemplare nella figura 3.1, sono le unità hardware utilizzate in questa tesi per svolgere tutti gli esperimenti. Sono state progettate dai laboratori *NEC* allo scopo di funzionare negli ambienti veicolari e comunicare tramite interfacce wireless con lo standard *IEEE 802.11p*. Possono essere utilizzate sia come *OBU* sia come *RSU*, però in questa tesi verrano entrambe utilizzate solamente come *OBU*.
L'attuale hardware di una *LinkBird-MX* è composto da un microprocessore *MIPS* a *64 bits*, *512 MB NAND-Flash*, *16 MB Nor-Flash* e *128 MB di SDRAM*. Supporta il sistema operativo *Linux*, attualmente con la versione del kernel 2.6.19. In aggiunta, prevede programmi di test e funzioni di diagnosi hardware. Con dimensioni fisiche di 153,5 millimetri(larghezza) 118 millimetri(profondità) 43mm(altezza), la *LinkBird-MX* risulta essere relativamente piccola ed ideale per l'uso come *OBU* di un veicolo.
L'industria garantisce un corretto funzionamento a temperature comprese tra -20 e +65 gradi centigradi. Inoltre, utilizza tecnologia per l'ibernazione per la riduzione dei tempi di avvio e dispone di un basso consumo di potenza (massimo 5W). Una *LinkBird-MX* è dotata di diverse interfacce. Oltre alle interfacce ampiamente utilizzate *RJ-45* e *USB*, viene anche equipaggiata con interfacce più avanzate, come le *VICS (Vehicle Information and Communication System)*, *UART integrata (Universal Asynchronous Receiver Transmitter)* e l'interfaccia *MOST (Media Oriented Systems Transport)*.
Per l'interfaccia wireless, la *LinkBird-MX*, seguendo un flessibile principio di progettazione, permette che possano essere collegate diverse schede di rete wireless. Nella prima versione verranno incluse una scheda basata su *miniPCI Chipset Atheros* ed un driver compatibile *IEEE 802.11p*. Per ulteriori estensioni sono a disposizione due slot per schede *PCMCIA*. Inoltre, sulla *LinkBird-MX* sono installati due connettori *RP-SMA* per le antenne. Le varie interfacce della

LinkBird-MX sono riassunte nella tabella 3.1.

Interfacce	Specifiche
Porta ethernet RJ-45	10/100Base-T 1 porta
USB	versione 2.0, 2 porte
VICS	1 porta
UART integrata	GPS,CAN,DSRC,232C
MOST	1 porta
Connettore ANT RP-SMA	2 porte
PCMCIA	2 slot
Mini-PCI: IEEE802.11p/a/b/g	1 slot
Power(DC12V)	1 porta

Tabella 3.1: Interfacce di una LinkBird-MX

La Resistenza alle vibrazioni dovute sia al motore sia al disturbo della strada è uno dei fattori più importanti per gli ambienti veicolari. La progettazione di una struttura debole può portare alla perdita della connessione del chip *IC* per il *PWB (Printed Wiring Board)*. Le *LinkBird-MX* sono state progettate bene contro tali vibrazioni. La piattaforma hardware esegue la pila del protocollo *NEC C2X*, che abilita la comunicazione wireless *ad-hoc* e *multi-hop* basata sul *Geocast*. Infine la tabella 3.2 mostra le principali caratteristiche dell'interfaccia di rete *IEEE 802.11p*.

Parametri	Dettagli
Frequenza/ Lunghezza Banda	5725MHz (145CH) 5925MHz (185CH) 10MHz / 20MHz
Versione	IEEE802.11p Draft 3.0, July 2007, software aggiornabile
Potenza di trasmissione	Massimo 21dBm
Bitrates	6, 9, 12, 18, 24, 36, 48, 54 Mb/s with 20MHz channel bandwidth 3, 4.5, 6, 9, 12, 18, 24, 27 Mb/s with 10MHz channel bandwidth
Operazioni multi canale	2 ricetrasmettitori operanti simultaneamente sui canali desiderati (ad esempio il canale 176 e 180 come in ETSI ITS-G5). Multiplexing, selezione del canale per-packet, assegnamento del canale basato sulla priorità, differenziazione del primo e dei seguenti hop grazie al NEC C2X-SDK.
Funzioni di controllo	Quando usato con il C2X-SDK ad ogni paccheto viene assegnato un diverso valore di potenza, rate, canale e categoria d'accesso ECDA 802.11. La priorità dei canali può essere assegnata dalle applicazioni.

Tabella 3.2: Caratteristiche dell'interfaccia di rete 802.11p

I laboratori *NEC* hanno sviluppato anche un pacchetto software denominato *C2X Software Development Kit* (*C2X SDK*). Questo SDK è un insieme di strumenti software che astrae la pila del protocollo per le comunicazioni *C2X* basate su tecnologie a corto raggio. Inoltre fornisce una *Application Programming Interface (API)* aperta che permette agli sviluppatori software di scrivere facilmente e velocemente le applicazioni per la sicurezza stradale, per le informazione sul traffico e per l'intrattenimento.
L'*SDK* è composto fondamentalmente da due moduli principali, la pila del protocollo di *C2X* e le *API C2X*. La pila del protocollo *C2X* abilita la comunicazione wireless *ad-hoc* e *multi-hop* tra veicoli e tra veicoli ed infrastrutture posizionate lungo le strade. Fornisce comunicazioni unicast, broadcast con ambito topologico e geografico. Algoritmi migliorati e meccanismi di protocollo assicurano un'efficiente ed affidabile trasmissione di dati in modo sicuro e forniscono anonimato ai nodi di comunicazione. Questo stack è stato sviluppato per il sistema operativo *Linux* e per le piattaforme hardware *x86*.
Le *API C2X* forniscono l'accesso alla pila del protocollo. Comprendono la definizione del formato dei messaggi, funzioni astratte per l'invio e la ricezione di tali messaggi e l'accesso alla gestione e alla configurazione delle interfacce. Le librerie possono aggregare i dati delle applicazioni e permettono lo scambio efficiente e strutturato delle informazioni tra i livelli del protocollo. Le *API* consentono alle applicazioni di essere eseguite su *AU* dedicate che possono essere eseguite su vari sistemi operativi.
Il *C2X SDK* distingue tra applicazioni *C2X* ed applicazioni *IP*. Le applicazioni *C2X* sono consapevoli del protocollo veicolare sottostante *C2X* e possono sfruttare e controllare i meccanismi del protocollo e le funzioni specifiche. Le applicazioni *IP* sono quelle applicazioni che si basano sul protocollo standard *IP* e per quanto riguarda la rete veicolare sfruttano il link di comunicazione trasparente.

3.1.2 Il Setup Fisico in Laboratorio

Le componenti principali e più importanti usate durante gli esperimenti nei laboratori sono le seguenti:

- computer portatile *Sony Vaio* con processore *Intel Core 2 Duo P7350* a *2,00 Ghz*, dotato di *3 Gb* di *Ram DDR2*, sistema operativo *linux Ubuntu 9.04* con *kernel 2.6.28-19*,
- computer portatile *Acer Travelmate* con processore *Intel Celeron 530* a *1,73 Ghz*, dotato di *1 Gb* di *Ram DDR2*, sistema operativo *linux Ubuntu 9.10* con *kernel 2.6.31-22*,

Figura 3.2: Antenna e *GPS* nella Fiat Panda

- due *LinkBird-MX* dotate di antenna wireless,
- due cavi *ethernet cross* per connettere le due *LinkBird-MX* ai due computer portatili,
- due lettori *GPS* connessi tramite periferiche *USB 2.0* ai due computer portatili,
- una cavo seriale per connettere tramite la porta seriale le *LinkBird-MX* ai computer portatili per modificare le impostazioni interne,
- altre componenti secondarie come prese elettriche ed altre ancora.

Non si richedono posizioni particolari per le due antenne delle *LinkBrid-MX*, mentre per l'uso dei due dispositivi per la lettura dei dati dal *GPS* è necessario che gli esperimenti si svolgano all'aria aperta senza la presenza di soffitti o grondaie che ostacolino la lettura dei dati dai satelliti.

3.1.3 Il Setup Fisico su Strada

Le componenti utilizzate negli esperimenti in laboratorio sono un sottoinsieme di quelle utilizzate negli esperimenti su strada. Infatti per effettuare le prove su strada sono necessarie tutte le componenti elencate sopra a cui vanno aggiunte

Figura 3.3: Antenna e *GPS* nella Pegeout 206

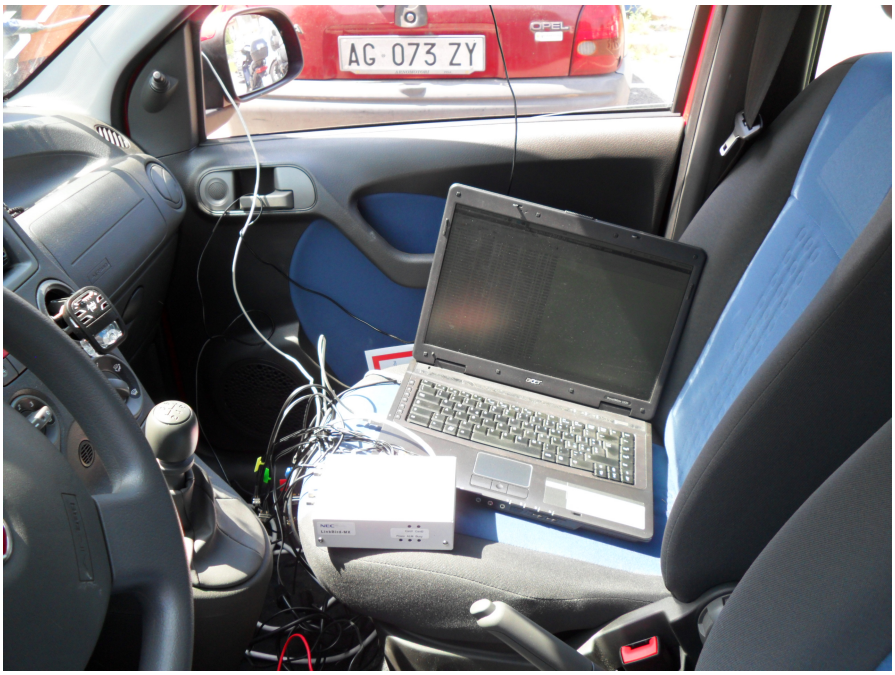

Figura 3.4: L'interno della Fiat Panda

Figura 3.5: L'interno della Pegeout 206

le seguenti che risultano assolutamente necessarie per lunghi esperimenti nelle vetture:

- veicolo *Fiat Panda* con presa accendisigari (figure 3.2 e 3.4),
- veicolo *Peugeot 206* con presa accendisigari (figure 3.3 e 3.5),
- due *inverter di potenza (Power Inverter)* da collegare alle auto tramite la presa per accendisigari e grazie alle quali è possibile fornire energia elettrica a tutti i componenti,
- altre componenti secondarie come prese elettriche ed altre ancora.

Va inoltre specificato che le antenne delle *LinkBird-MX* vengono fissate sul tetto della macchina grazie alla loro base magnetica per una migliore ricezione ed invio ed i dispositivi per la lettura GPS vengono posizionati all'esterno dei veicoli fissandoli con del nastro adesivo.

3.2 Il Simulatore

Il Simulatore è un componente software fondamentale il cui compito è quello di simulare una conversazione VoIP tra due nodi inviando e ricevendo pacchetti

IPv4. Per far ciò invia e riceve datagrammi *UDP* interfacciandosi alle *LinkBird-MX* tramite connessione *Ethernet*. Poiché siamo interessati solamente alla fase di invio e di ricezione per valutare perdite ed altri parametri e non del contenuto vocale, il payload dei pacchetti inviati sarà composto da byte casuali, di cui i primi 4 contenenti l'identificativo del pacchetto.
Inoltre, poiché siamo principalmente interessati a studiare l'invio e la ricezione su veicoli in movimento, il simulatore si occupa di interrogare, tramite un *Timer*, ogni 100 millisecondi il *GPS* leggendo e gestendo i relativi dati. La gestione del *GPS* consiste nell'aspettare che il dispositivo si connetta ad uno o più satelliti, sondarlo costantemente ed elaborare i dati ricevuti. Va inoltre notato che il dispositivo *GPS* aggiorna i propri dati dopo ogni secondo, quindi circa dieci letture consecutive risulteranno identiche. I dati restituiti dal *GPS* che ci interessano maggiormente sono:

- il *tempo*: indica il tempo rilevato durante l'ultima acquisizione dei dati,
- la *longitudine*: viene misurata in gradi sessagesimali ed indica la longitudine del veicolo rilevata durante l'ultima acquisizione dei dati,
- la *latitudine*: viene misurata in gradi sessagesimali ed indica la latitudine del veicolo rilevata durante l'ultima acquisizione dei dati,
- la *velocità*: viene misurata in metri al secondo ed indica la velocità del veicolo rilevata durante l'ultima acquisizione dei dati.

Le modalità e le tempistiche dell'invio e della ricezione dei pacchetti dipendono da vari parametri che l'utente deve settare tramite l'interfaccia grafica. Vediamo ora dettagliatamente questi parametri e in che modo essi influenzano e caratterizzano la simulazione:

- il periodo di *parlato medio*: questo parametro, espresso in millisecondi, esprime quanto dovrebbe essere il periodo medio durante una conversazione in cui si parla attivamente, e di conseguenza, per i nostri scopi, si inviano pacchetti,
- il periodo di *silenzio medio*: al contrario rispetto a quello precedente, questo parametro, espresso in millisecondi, esprime quanto dovrebbe essere il periodo medio durante una conversazione in cui non si parla, e di conseguenza, per i nostri scopi, non si inviano pacchetti,
- il periodo di *parlato minimo*: questo parametro, espresso in millisecondi, impone un vincolo sul minimo valore che può assumere un periodo di parlato per rendere più realistica la simulazione,

- la *porta IPv4*: rappresenta la porta utilizzata per indirizzare i pacchetti,
- l'*indirizzo di destinazione*: rappresenta l'indirizzo del nodo destinazione ed ha la classica forma *aaaa.bbbb.cccc.dddd*,
- la *durata dell'esperimento*: questo parametro, espresso in millisecondi, indica per quanto tempo si svolgerà la simulazione,
- la *frequenza di invio (bit-rate)*: questo parametro, espresso in bit/s, indica quanti kilobit devono essere spediti nell'arco di un secondo, e quindi influenza la frequenza di invio,
- la *dimensione dei pacchetti*: questo parametro, espresso in Byte, indica la dimensione del payload dei pacchetti *IPv4* che verranno inviati, e quindi anch'esso influenza la frequenza di invio,
- il *percorso*: indica la cartella dove verranno salvati i file di log generati dal simulatore.

Vediamo un'esempio delle prime due righe di due file di log, uno di ricezione uno di invio, riguardanti una stessa simulazione:

```
Prime due righe del file di log di invio:
64000 2400000 1506 2000 1500 250 80
1 175036036620 1304068903.00 43.71727900 10.42939300 0.000 53.2800

Prime due righe del file di log di ricezione:
64000 2500000 1506 2000 1500 250 80
1 129419764180 1304068903.00 43.71732130 10.42926280 0.000 25.9000
```

Ogni simulazione, sia di invio sia di ricezione, porta alla creazione di un file di log molto importante per l'analisi dei risultati e per la fase di Post-Processing. I file di log relativi agli invii e quelli relativi alle ricezioni hanno un preciso formato e sono strutturati nello stesso modo. La prima riga di un file di log contiene i parametri della simulazione, cioè la frequenza di invio, la durata dell'esperimento, la porta *IPv4*, il silenzio medio, il parlato medio, il parlato minimo e la dimensione del pacchetto separati da uno spazio.
Dopo la prima riga seguono tante righe quanti sono stati i pacchetti inviati o ricevuti, ognuna delle quali contenente, separati da spazi, l'identificativo del pacchetto, il tempo in nanosecondi in cui si è verificato l'invio o la ricezione ed i dati raccolti dal *GPS*. Qualsiasi file di log che non abbia questo formato non viene ritenuto ben formato e non verrà processato correttamente dal programma di Post-Processing.

Una parte fondamentale del simulatore è come decide di simulare la conversazione che è caratterizzata dall'alternanza di periodi di parlato a periodi di silenzio variabili e non predicibili a priori data la natura aleatoria di questi valori. Studi al riguardo hanno dimostrato come questi periodi siano modellabili correttamente tramite la funzione di distribuzione esponenziale con parametro $\lambda > 0$ ([16]).
Nella teoria delle probabilità la distribuzione esponenziale è una distribuzione di probabilità continua che descrive eventi che occorrono continuamente ed indipendentemente ad una frequenza media costante (assenza di memoria). La distribuzione esponenziale con parametro $\lambda > 0$, ha funzione di densità di probabilità definita sui numeri reali positivi R^+ pari alla funzione esponenziale $f(x) = \lambda e^{-\lambda x}$. Inoltre una variabile aleatoria X con distribuzione esponenziale di parametro $\lambda > 0$ ha funzione di ripartizione $F(x) = P(X \leq x) = 1 - e^{-\lambda x}$. Infine possiamo caratterizzare la funzione inversa della distribuzione esponenziale come $F^{-1}(x) = \frac{-\log(1-x)}{\lambda}$ dove x è una probabilità.
Vediamo quindi come avviene concretamente il calcolo del periodo del parlato (per il periodo di silenzio la procedura è speculare). Per prima cosa si prende il parametro *media del parlato (media)*, che non deve essere confuso con il parametro λ, e si ottiene da esso $\lambda = \frac{1}{media}$. A questo punto si calcola un numero casuale compreso tra 0 ed 1 che rappresenterà la nostra x e si calcola la funzione inversa fornendo i parametri λ ed x. Il valore restituito dalla funzione inversa è direttamente il periodo di parlato che stavamo cercando.
Rispetto al periodo di silenzio, il periodo di parlato ha un vincolo in più, e cioè quello di essere superiore ad un certo valore di soglia (il parametro *parlato minimo*) perché la distribuzione esponenziale può generare valori anche molto piccoli rispetto alla media che renderebbero poco credibile la simulazione. Una volta ottenuti i periodi di parlato e silenzio con dei semplici *Timer* è possibile gestire il comportamento del simulatore.
Concludiamo descrivendo come il simulatore gestisce la frequenza di invio dei dati. I parametri fondamentali a questo scopo sono due : la *frequenza di invio (bit-rate)* e la *dimensione dei pacchetti (size)*, infatti il primo dice quanti kilobit dobbiamo inviare nell'arco di un secondo, mentre il secondo dice quanto grandi sono i pacchetti. Modificare uno solo o entrambi questi parametri implica modificare le modalità di invio dei pacchetti e di conseguenza il tempo che intercorre tra un invio ed un altro. Questo tempo che intercorre tra due invii viene dunque calcolato tramite la formula $\frac{bit-rate}{size*8}$.
Quindi si prende il *bit-rate*, espresso in kilobit/s, e la dimensione del pacchetto, espressa in byte, che viene moltiplicata per 8 ottenendo così la dimensione del pacchetto in bit, e si dividono ottenendo così il tempo necessario per inviare tali bit. Questo valore viene ulteriormente controllato e trattato per capire se

è necessario un tempo dell'ordine dei millisecondi o un tempo dell'ordine dei nanosecondi. Basta dunque sospendere l'invio dei pacchetti per tale periodo per gestire questo aspetto della simulazione.
Opzionalmente, in parallelo con la vera e propria simulazione, è possibile avviare il beaconing. Con il termine beaconing, in questo contesto, intendiamo l'invio di messaggi di segnalazione in contemporanea all'invio dei pacchetti VoIP sullo stesso canale. Il beaconing viene utilizzato per *infastidire* l'applicazione e simulare una situazione in cui un'applicazione diversa concorra per l'uso del canale. Va altresì fatto notare che la scelta di un'applicazione di beaconing è anche forzata dal fatto che queste sono applicazioni fondamentali per gli ambienti veicolari ed in particolar modo per le applicazioni di sicurezza, dunque in una situazione reale qualsiasi applicazione dovrà sempre concorrere con applicazioni di questo tipo. Quindi, non essendo interessati al contenuto dei pacchetti, i messaggi di beaconing vengono spediti sotto forma di messaggi *Single-Hop-Broadcast (SHB)* e non contengono informazioni utili. Se attivato il beaconing viene influenzato dai parametri:

- l'ID del veicolo: indica l'identificativo univoco del veicolo che sta inviando i messaggi di beaconing,
- la dimensione dei messaggi di beaconing: questo parametro, espresso in Byte, indica la dimensione del payload dei pacchetti SHB di beaconing che verranno inviati,
- l'intervallo: questo parametro, espresso in millisecondi, indica l'intervallo temporale tra l'invio di due messaggi *SHB* di beaconing.

Più dettagliatamente il beaconing viene implementato avviando un thread parallelo che si occupa dell'invio (per il dispositivo inviante) o della ricezione (per il dispositivo ricevente) dei messaggi *SHB*. Tali messaggi contengono l'identificativo del veicolo, un numero intero indicante l'identificativo del messaggio stesso, e, se presenti, i dati *GPS*. Tramite un *Timer* si gestisce l'intervallo di invio. Se il beaconing viene attivato viene generato anche un file di log contenente l'identificativo del messaggio con i dati *GPS* se presenti per ogni messaggio di beaconing inviato o ricevuto.

3.3 Il programma di Post-Processing

Il programma che gestisce il Post-Processing è l'altra componente fondamentale di questa tesi e, in ultima analisi, si occupa di analizzare due file di log, uno di invio ed uno di ricezione, generati da uno stesso esperimento, e di estrarre da essi altri due file di log che diano una maggiore informazione sugli esperimenti

avvenuti. I due file di log di input devono avere una precisa struttura (come descritto nel precedente paragrafo), altrimenti il programma restituirà dei messaggi di errore. Prima di vedere ed analizzare i due file di log di output che vengono generati da questo processo (anch'essi con una ben definita struttura), è necessario capire e descrivere dettagliatamente ciò che fa il programma di Post-Processing.
L'esecuzione di un'istanza del programma di Post-Processing comincia con l'apertura dei due file di log di input e con la creazione e susseguente apertura dei due file di log di output, ed il tutto deve avvenire ovviamente senza alcun tipo di errore, pena l'uscita dal programma. Una volta aperti i file di log di input si prende la loro prima riga contenente i dati dell'esperimento, si salvano questi dati e si confrontano i valori relativi alla dimensione del pacchetto e della porta *IPv4* perché questi due parametri devono combaciare in una stessa simulazione. Se le prime righe dei file di log di input sono ben formate e i due parametri combaciano allora è possibile iniziare a scrivere sul primo file di log di output, quello cioè che conterrà le generiche informazioni sul Post-Processing, i percorsi dei due file di log di input e la cartella dove verranno generati i due file di log di output.
A questo punto si iniziano a leggere le varie righe dei due file di log di input per calcolare i vari parametri che ci interessano come la distanza, il jitter, il throughput ed altri. Poiché, a seconda della durata della simulazione, i file di log possono essere molto grandi (anche dell'ordine dei megabyte) con un numero molto alto di righe, il programma scandisce i file una sola volta e non agisce su ulteriori strutture dati complesse operando così in $O(n)$ dove n rappresenta il numero di pacchetti inviati durante la simulazione. Questa ottimizzazione permette di processare file molto grandi in pochissimi secondi anche con dispositivi dotati di poca potenza di calcolo.
Quindi si parte con la lettura in entrambi i file di una riga e si continua finché uno dei due file o entrambi non sono finiti aggiornando ad ogni lettura il numero di righe lette sia nel file di log di invio sia in quello di ricezione. Ogni riga correttamente letta conterrà sicuramente l'id del pacchetto inviato/ricevuto ed il tempo espresso in nanosecondi in cui tale pacchetto è stato inviato/ricevuto. Inoltre accanto a questi due dati, se il *GPS* ha restituito correttamente i dati letti, compariranno il tempo *GPS*, la latitudine, la longitudine, la velocità e la direzione relative al momento in cui il pacchetto è stato inviato/ricevuto. Va notato dunque che ogni riga dei file di log conterrà o due o sette valori separati da uno spazio bianco, se ciò non avviene allora il file non è considerato ben formato.
A questo punto abbiamo due righe dei file di log, una riga relativa all'invio e l'altra relativa alla ricezione e per prima cosa si confrontano gli id, ottenendo

due possibili casi. Il caso in cui l'id del pacchetto inviato è minore dell'id del pacchetto ricevuto ci fa capire che uno o più pacchetti non sono stati ricevuti e quindi non è possibile calcolare i dati che vogliamo. Quindi in questo caso si continua la lettura delle righe nel file di log di invio (mentre la lettura nel file di log di ricezione si interrompe temporaneamente) finché non ne troviamo una che ha come id lo stesso valore della riga del file di ricezione. Ad ogni lettura che indica una non corrispondenza tra gli id si incrementa un'apposita variabile che conta il numero dei pacchetti persi. Invece il caso in cui i due id corrispondono denota che all'invio del pacchetto con tale id è susseguita la corretta ricezione di tale pacchetto. Quindi si leggono i valori rimanenti delle due righe e si cerca di calcolare il jitter, la distanza e la velocità relative a queste due righe (per i dettagli dei calcoli vedere i paragrafi 3.5.1, 3.5.2, 3.5.3, 3.5.4 e 3.5.5). Fatto ciò si continua la lettura delle righe avanzando in entrambi i file.
Parallelamente a questa scansione delle righe dei file di log di input si ha un ulteriore controllo sui valori letti al fine di generare il secondo file di log di output, quello cioè che verrà usato da programmi esterni per visualizzare grafici relativi agli esperimenti e per capire se c'è correlazione tra i parametri calcolati. Quindi questo file di log che verrà generato sarà strutturato in colonne, la prima riga conterrà i nomi dei parametri separati da uno spazio, e le seguenti righe conterranno i dati relativi. Data la grande mole di dati e le esigenze di visualizzare correttamente i grafici, i valori ottenuti vengono divisi in intervalli ognuno dei quali ha durata di un secondo, quindi ogni riga del file contiene i valori medi collezionati nell'arco di un secondo. Per far ciò ci viene in aiuto il tempo *GPS*, infatti questo scatta proprio dopo un secondo, cosa che agevola notevolmente questo compito. Quindi ogni volta che nel file di log di invio cambia il tempo *GPS*, si calcola il throughput che c'è stato in questo secondo, le medie dei valori che ci interessano in questo lasso di tempo ed altri parametri e si aggiunge una riga al secondo file di log di output.
Vengono presi in considerazione i tempi *GPS* del log di invio perché siamo sicuri che non ci siano perdite in questo log, e quindi in questo modo non avremo sbalzi del tempo *GPS* dovuti a smarrimenti di pacchetti. Va comunque notato che ci saranno ugualmente sbalzi dei tempi *GPS* nel log di invio dovuti ai periodi di silenzio (che sono tipicamente dell'ordine dei secondi), ma questo non rappresenta un problema, visto che per ogni secondo di silenzio si aggiunge una riga con i valori azzerati per sottolineare che in quel determinato intervallo non sono stati inviati e ricevuti pacchetti. Allo scattare di un nuovo intervallo si azzerano i valori per i nuovi calcoli. Più precisamente le colonne di questo file di log di output sono:

- *Time*: indica il numero progressivo degli intervalli e di conseguenza indica

il passare del tempo infatti il numero totale degli intervalli indica la durata della simulazione in secondi,

- *Jitter*: indica il jitter medio, calcolato in millisecondi, che c'è stato nell'arco di quel secondo,
- *Distance*: indica la distanza media tra i veicoli, calcolata in metri, che c'è stata nell'arco di quel secondo,
- *VelocityS*: indica la velocità media del dispositivo che inviava pacchetti, calcolato in metri/secondo, che c'è stata nell'arco di quel secondo,
- *VelocityR*: indica la velocità media del dispositivo che riceveva pacchetti, calcolato in metri/secondo, che c'è stata nell'arco di quel secondo,
- *ThroughputS*: indica il throughput medio del dispositivo che inviava pacchetti, calcolato in bit/secondo, che c'è stato nell'arco di quel secondo,
- *ThroughputR*: indica il throughput medio del dispositivo che riceveva pacchetti, calcolato in bit/secondo, che c'è stato nell'arco di quel secondo,
- *RapportoT*: indica il rapporto tra il throughput del dispositivo che riceveva i pacchetti ed il throughput del dispositivo che inviava pacchetti che c'è stato nell'arco di quel secondo (il valore 1 indica lo stesso throughput nei due dispositivi),
- *Inviati*: indica il numero di pacchetti inviati nell'arco di quel secondo,
- *Ricevuti*: indica il numero di pacchetti ricevuti nell'arco di quel secondo,
- *Perdite*: indica il numero di pacchetti persi nell'arco di quel secondo,
- *Rapporto*: indica il rapporto dei pacchetti inviati nell'arco di quel secondo (il valore 1 indica nessun pacchetto perso mentre il valore 0 indica che tutti i pacchetti sono persi),

Arrivati alla fine della lettura delle righe dei file di log di input, vengono scritte ulteriori informazioni sul primo file di log di output relative all'intero processo di Post-Processing, e cioè:

```
Post-Processing:
- Log di Invio:  ···
- Log di Ricezione:  ···
- Cartella di Output:  ···

Dati dell'esperimento:
```

```
- Frequenza: ···
- Porta IPv4: ···
- Dimensione Pacchetto: ···
- Minimo Periodo di Parlato: ···
- Durata Esperimento: ···
- Periodo di Parlato Medio: ···
- Periodo di Silenzio Medio: ···

Dettagli dell'esperimento:

Jitter:
- Jitter Minimo: ···
- Jitter Massimo: ···
- Jitter Medio: ···

Distanza:
- Distanza Minima: ···
- Distanza Massima: ···
- Distanza Media: ···

Throughput:
- Throughput Sender Minimo: ···
- Throughput Sender Massimo: ···
- Throughput Sender Medio: ···
- Throughput Receiver Minimo: ···
- Throughput Receiver Massimo: ···
- Throughput Receiver Medio: ···
- Throughput Receiver Medio Reale: ···
- Rapporto Throughput: ···

Velocità:
- Velocità Sender Minima: ···
- Velocità Sender Massima: ···
- Velocità Sender Media: ···
- Velocità Receiver Minima: ···
- Velocità Receiver Massima: ···
- Velocità Receiver Media: ···

Pacchetti:
- Pacchetti Inviati: ···
```

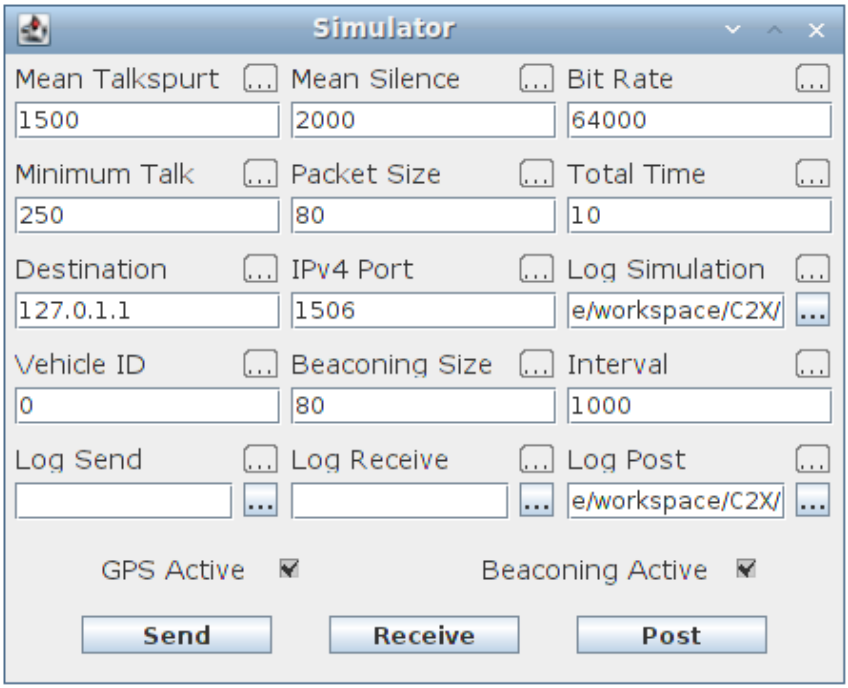

Figura 3.6: L'interfaccia Utente

- `Pacchetti Ricevuti:` ···
- `Pacchetti Persi:` ···
- `Rapporto Medio:` ···
- `Numero intervalli vuoti:` ···
- `Massimo intervallo vuoto:` ···

In conclusione abbiamo che il programma di Post-Processing, se avviato su due file di log di input corretti e ben formati, genera due file di log di output, uno verboso contenente generiche informazioni sui risultati ed uno che sarà destinato ad essere usato in programmi esterni per la visualizzazione di grafici.

3.4 L'Interfaccia Grafica

L'interfaccia utente o *Graphical User Interface (GUI)*, mostrata in figura 3.6, facilita il compito dell'utente permettendogli di impostare i vari parametri, di avviare una simulazione o il Post-Processing e avvertendolo dei risultati ottenuti. Poiché tipicamente dopo una simulazione segue il relativo Post-Processing l'interfaccia incorpora sia i parametri e pulsanti relativi alla simulazione, sia i parametri e i pulsanti relativi al Post-Processing.

L'interfaccia, semplice ed intuitiva, è stata realizzata col *framework Swing* per *Java* facente parte delle *Java Foundation Classes (JFC)* ed orientato proprio allo sviluppo di interfacce grafiche. I parametri, modificabili tramite delle semplici *textbox*, sono esattamente gli stessi elencati e descritti nei paragrafi precedenti e ad ognuno di loro è associata un'etichetta indicante il nome del parametro ed una breve *tooltip* che lo descrive. Le azioni possibili sono le seguenti:

- Invio (tasto *Send*): questo tasto permette l'inizio di una simulazione in modalità invio per un prefissato periodo di tempo, ad un preciso indirizzo e porta *IPv4* in cui i pacchetti hanno tutti una determinata dimensione e vengono inviati con una precisa frequenza. Inoltre i periodi di parlato e silenzio della simulazione hanno una precisa media e viene fissato un minimo periodo di parlato. Infine l'utente può settare dove verrà generato il file di log relativo alla simulazione. Di conseguenza i parametri che influenzano questa azione sono: *Mean Talkspurt, Mean Silence, Bit Rate, Minimum Talk, Packet Size, Total Time, Destination, IPv4 Port e Log Simulation.* Se viene attivato il beaconing, allora questa azione viene influenzata anche dai parametri: *Vehicle ID, Beaconing Size e Interval*

- Ricezione (tasto *Receive*): questo tasto permette l'inizio di una simulazione in modalità ricezione per un prefissato periodo di tempo, su di una precisa porta *IPv4* in cui i pacchetti hanno tutti una determinata dimensione. Inoltre l'utente può settare dove verrà generato il file di log relativo alla simulazione. Di conseguenza i parametri che influenzano questa azione sono: *Packet Size, Total Time, IPv4 Port e Log Simulation.*

- Post-Processing (tasto *Post*): questo tasto permette l'inizio del Post-Processing tra due file di log, uno di invio ed uno di ricezione, determinati dall'utente. Inoltre l'utente può settare dove verranno generati i file di log relativi al Post-Processing. Di conseguenza i parametri che influenzano questa azione sono: *Log Send, Log Receive e Log Post.*

Tramite l'intefaccia utente è inoltre possibile scegliere se attivare la lettura dei dati dal dispositivo *GPS* e se attivare il beaconing. Ad ogni azione compiuta tramite l'interfaccia utente viene mostrato un messaggio che informa l'utente sull'esito della suddetta azione. La figura 3.7 mostra alcuni esempi di tali messaggi: il caso (a) mostra il messaggio che viene visualizzato quando una simulazione, di invio o ricezione, viene eseguita correttamente; il caso (b) mostra un esempio di errore riscontrato durante una simulazione di invio, più precisamente è stato inserito un indirizzo *IPv4* non corretto; il caso (c) mostra il messaggio che viene visualizzato quando il Post-Processing viene eseguito cor-

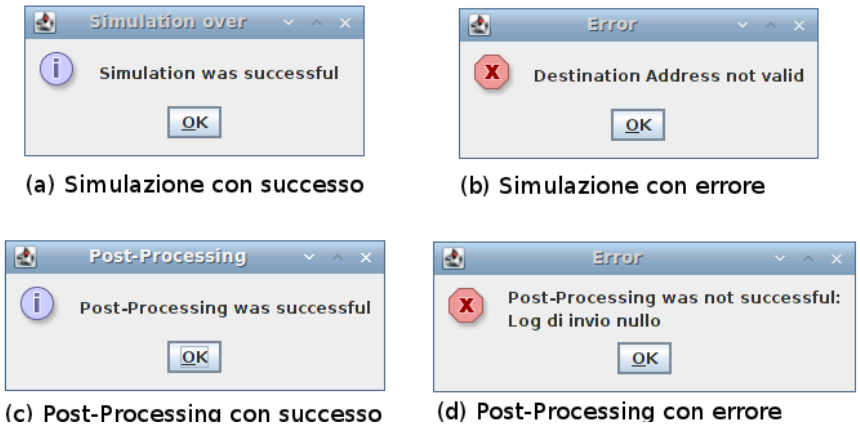

Figura 3.7: Esempi di messaggi dell'interfaccia utente

rettamente; infine il caso (d) mostra un esempio di errore riscontrato durante il Post-Processing, più precisamente non è stato specificato il file di log di invio.

3.5 I parametri

La fase di Post-Processing si incentra sul calcolo e sull'analisi di alcuni parametri che ci interessano particolarmente. Lo studio di questi parametri e le modalità con cui vengono calcolati sono fondamentali per una corretta interpretazione dei risultati ottenuti.

3.5.1 Il Jitter

Come indicato nell'*RFC 3393* ([12]), la variazione del ritardo dei pacchetti viene talvolta chiamata *jitter*. Questo termine, tuttavia, genera confusione perché viene utilizzato in modi diversi e da diversi gruppi di persone. *Jitter* ha tipicamente due significati: il primo significato è la variazione di un segnale rispetto a qualche segnale di clock, in cui il tempo di arrivo del segnale ci si aspetta coincida con l'arrivo del segnale di clock. Questo significato viene utilizzato con riferimento a segnali sincroni. Il secondo significato ha a che fare con la variazione di una metrica (ad esempio, il ritardo) rispetto a qualche altra metrica di riferimento (ad esempio, ritardo medio o ritardo minimo).
Proprio per questo uso, a volte improprio, del termine *jitter*, sarebbe più corretto chiamarlo *IP Packet Delay Variation (ipdv)* per una maggior chiarezza (anche se noi continueremo a chiamarlo *jitter* per comodità). L'*ipdv* di una coppia di pacchetti all'interno di un flusso di dati è definito per una coppia di pacchetti selezionata nel flusso che va dalla misurazione in un punto *MP1* al-

la misurazione in un punto *MP2*. L'*ipdv* è la differenza tra il ritardo *one-way* dei pacchetti selezionati. Un uso importante dell'*ipdv* è il dimensionamento dei buffer per le applicazioni che richiedono la trasmissione periodica di pacchetti (per esempio, voce o video). Ciò che è normalmente importante in questo caso è la variazione di ritardo massimo, che viene utilizzata per dimensionare i buffer per questo tipo di applicazioni. Altri usi di questo parametro sono, ad esempio, la gestione di code all'interno di una rete (o router).
Tipicamente calcolare un parametro di questo tipo implica avere i clock dei dispositivi sincronizzati, ma questa, purtroppo, risulta essere un'operazione non facile e molto prona agli errori. Per questo motivo si è deciso di calcolare il *jitter* senza avere i clock sincronizzati. Ci viene in aiuto per questo scopo la definizione di *jitter* in *RTP* [13] che definisce il *jitter J* come la deviazione media della *differenza D* di spaziatura tra i pacchetti del mittente e i pacchetti del destinatario. Siano *Si* ed *Sj* i tempi in cui i pacchetti i e j vengono inviati (il pacchetto j deve essere successivo al pacchetto i), e siano inoltre *Ri* ed *Rj* i tempi in cui i pacchetti i e j vengono ricevuti. Allora possiamo caratterizzare la *distanza* tra i e j con la formula:

$$D(i,j) = (Rj - Ri) - (Sj - Si) = (Rj - Sj) - (Ri - Si).$$

Una volta calcolata la *distanza* tra i e j è possibile calcolare effettivamente il valore del *jitter* al passo i in funzione del valore del *jitter* al passo *i - 1*, ed è possibile farlo tramite la formula:

$$J(i) = J(i-1) + (|D(i-1,i)| - J(i-1))/16.$$

Si noti che i pacchetti i e j non devono essere necessariamente consecutivi ed il parametro 1/16 fornisce una buona riduzione del rumore.

3.5.2 La Distanza

La distanza tra i due veicoli in movimento viene misurata in metri e calcolata sfruttando i dati sulle posizioni raccolti dal dispositivo *GPS*. Poiché le posizioni dei veicoli vengono espresse tramite longitudine e latitudine in gradi sessagesimali, per ricavare la distanza in metri è necessaria un'ulteriore elaborazione di questi dati. Indicando con $l1$ e $L1$ rispettivamente la latitudine e la longitudine del primo punto, e con $l2$ e $L2$ rispettivamente la latitudine e la longitudine del secondo punto, allora abbiamo che la distanza (d) tra i due punti espressa in metri è data dalla formula:

$$d = 2 \cdot R \cdot \cos(l1) \cdot \cos(l2) \cdot \arcsin(\sqrt{\sin^2(\tfrac{l1-l2}{2}) + \cos(l1) \cdot \cos(l2) \cdot \sin^2(\tfrac{L1-L2}{2})})$$

dove $R = 6372797.6$ metri è il raggio medio della terra. Va inoltre sottolineato che le latitudini e le longitudini in questa formula sono espresse in radianti, quindi per convertire una misura espressa in gradi sessagesimali in radianti bisogna moltiplicarla per π e dividere il risultato per 180.

3.5.3 La Velocità

La velocità dei veicoli viene rilevata per entrambi i dispositivi e viene misurata in metri al secondo. I valori, sia per quanto riguarda il dispositivo che invia sia per quanto riguarda il dispositivo che riceve, vengono estratti direttamente dalle tracce dei file di log. Infatti la velocità è uno dei parametri letti direttamente dal dispositivo *GPS*, e quindi non necessita di ulteriori elaborazioni. Se si è interessati ad esprimere questi valori in chilometri all'ora è sufficiente moltiplicare il valore espresso in metri al secondo per 3600 (in modo da ottenere un valore espresso in metri all'ora) e dividere il risultato ottenuto per 1000.

3.5.4 Il Throughput

Il throughput di un link (canale) di comunicazione può essere definito come la sua capacità di trasmissione effettivamente utilizzata. Il throughput non deve essere confuso con la capacità del link, infatti sia la capacità che il throughput si esprimono in bit/s, ma mentre la prima esprime la frequenza trasmissiva massima alla quale i dati possono viaggiare, il throughput è un indice dell'effettivo utilizzo della capacità del link. Il throughput può anche essere definito come la quantità di dati trasmessi in una unità di tempo, il secondo, e dipende esclusivamente da quanta informazione viene immessa sul canale nella trasmissione. Ovviamente il throughput è sempre inferiore rispetto alla banda massima disponibile.
Calcolare il throughput nel nostro caso risulta molto semplice, infatti calcolandolo per ogni intervallo della durata di un secondo, è sufficiente moltiplicare il numero di pacchetti inviati nell'arco di tale intervallo per la dimensione del pacchetto (essendo la dimensione del pacchetto espresso in byte questa dovrà essere moltiplicata per 8). Confrontare il throughput del dispositivo che invia e del dispositivo che riceve ci aiuta ulteriormente a capire le dinamiche degli invii e delle ricezioni tenendo conto anche delle pause di silenzio tipiche del *VoIP* che vanno inevitabilmente ad influire sui troughput. Viene inoltre calcolato il throughput del ricevente al netto degli intervalli nulli. Questo parametro viene calcolato escludendo gli intervalli in cui non è stato ricevuto nessuno dei pacchetti invia-

ti, e viene preso in considerazione per valutare l'andamento del throughput nei periodi in cui il sistema di comunicazione funziona perfettamente.

3.5.5 Le Statistiche sui pacchetti

Conoscere con precisione le statistiche riguardanti i pacchetti inviati e ricevuti durante una simulazione permette sicuramente una migliore conoscenza degli esperimenti svolti. Per questo motivo, vengono tenuti in considerazione ulteriori parametri riguardanti i pacchetti per ogni intervallo da un secondo:

- Pacchetti inviati: indica il numero di pacchetti correttamente inviati dal dispositivo che invia. Viene calcolato incrementando un contatore ogni volta che un invio va a buon fine.
- Pacchetti ricevuti: indica il numero di pacchetti correttamente ricevuti dal dispositivo che riceve. Viene calcolato incrementando un contatore ogni volta che un pacchetto viene ricevuto.
- Pacchetti persi: indica il numero di pacchetti che non sono giunti a destinazione. Viene calcolato incrementando un contatore ogni volta che si perde un pacchetto, cioè ogni volta che non c'è corrispondenza di *id* nei file di log.
- Rapporto: indica il rapporto di invio dei pacchetti dove il valore 1 indica che tutti i pacchetti inviati sono stati correttamente ricevuti, mentre il valore 0 indica, al contrario, che nessuno dei pacchetti inviati è stato ricevuto. Viene calcolato dividendo il numero di pacchetti ricevuti con il numero dei pacchetti inviati,
- Numero intervalli vuoti: indica il numero di intervalli in cui il dispositivo trasmittente ha inviato almeno un pacchetto e il dispositivo ricevente non ha ricevuto alcun pacchetto,
- Massimo intervallo vuoto: indica il massimo numero di intervalli consecutivi in cui il dispositivo ricevente non ha ricevuto nessuno dei pacchetti inviati.

Gli ultimi due parametri sono particolarmente importanti per il *VoIP*, infatti sapere quanti sono gli intervalli in cui non si riceve niente ed il massimo intervallo continuo di non ricezione ci aiuta a capire quanti problemi ci saranno durante la comunicazione. Infatti sapendo ad esempio che per 5 secondi consecutivi non è stato ricevuto alcun pacchetto, sappiamo che la conversazione in quei 5 secondi non è stata riportata correttamente. Avere troppi intervalli nulli implica quindi l'impossibilità di effettuare una chiamata correttamente.

Capitolo 4

I risultati degli esperimenti

Questo capitolo descrive dettagliatamente gli esperimenti eseguiti ed i relativi risultati ottenuti. Verranno presentati qua solo una piccola parte dei dati raccolti che risultano particolarmente significativi per i nostri scopi. Inoltre i dati raccolti ed analizzati verranno presentati anche in forma tabellare e di grafico per una maggiore chiarezza e per mostrare le relazioni che intercorrono tra alcuni parametri presi in considerazione. I parametri accompagnati da (I) sono relativi al dispositivo inviante, mentre i parametri accompagnati da (R) si riferiscono al dispositivo ricevente. Verranno descritti brevemente gli esperimenti svolti in laboratorio mentre verrà data maggiore enfasi agli esperimenti svolti su strada.

4.1 Gli esperimenti in laboratorio

Questa tipologia di esperimenti, svolti presso i laboratori del *Consiglio Nazionale delle Ricerche (C.N.R.) di Pisa*, ha avuto lo scopo di testare la corretta comunicazione delle due *LinkBird-MX*, usate durante gli esperimenti, e di testare il funzionamento del sistema in una situazione vantaggiosa senza la presenza di ostacoli tra i due dispositivi e senza la gestione del loro movimento. In definitiva questi esperimenti sono stati svolti aggirando i tipici problemi caratteristici delle comunicazioni wireless, come ad esempio la presenza di oggetti ostacolanti tra le antenne, a fronte di una maggiore attenzione sulla corretta comunicazione dei dispositivi in previsione delle prove su strada.
Tutti gli esperimenti di questo tipo sono stati svolti posizionando i dispositivi vicini, soprattutto le antenne, e senza alcun tipo di ostacolo tra loro. Fatto ciò sono stati avviati molti e vari esperimenti di cui non riportiamo i risultati perché poco importanti per i nostri scopi. Va ugualmente notato che in ogni esperimento di questo tipo tutti i pacchetti inviati venivano correttamente rice-

Figura 4.1: Mappa con il tratto di strada percorsa [24]

vuti senza ritardi significativi.
Più precisamente questi esperimenti sono stati svolti in tre mandate: una prima mandata di esperimenti è stata svolta dentro un laboratorio al coperto per testare il sistema nel suo stato iniziale, invece la seconda mandata si è svolta all'esterno dell'edificio per testare anche il dispositivo per la lettura dei dati *GPS*, infine la terza mandata è svolta all'esterno dell'edificio per testare il dispositivo per la lettura dei dati *GPS* insieme al beaconing. Anche la lettura dei dati *GPS* ha sempre dato esito positivo. Tutti questi esperimenti sono stati svolti configurando le *LinkBird-MX* in modo che comunicassero tra loro sul canale di controllo a 3 Mbit/s spedendo messaggi di 80 byte ad una frequenza di 64000 bit/s.

4.2 Gli esperimenti su strada

Veniamo ora alla parte centrale di questa tesi, cioè la descrizione e la presentazione dei risultati degli esperimenti svolti su strada. A causa del gran numero di esperimenti svolti, e di conseguenza della gran mole di dati raccolta, verranno qua presentati solo quelli di maggior interesse per i nostri scopi. Come già accennato nel precedente capitolo, questi esperimenti sono stati svolti percorrendo un tratto stradale della città di *Pisa* adiacente all'area di ricerca del C.N.R. con due veicoli predisposti appositamente per gli esperimenti, uno adibito all'invio e l'altro adibito alla ricezione. La figura 4.1 mostra la mappa contenente il tratto di strada percorsa.
Nella versione standard degli esperimenti il veicolo inviante viaggiava sempre davanti rispetto a quello ricevente, ma ci sono stati casi in cui i due veicoli si sono invertiti, e questi casi verranno esplicitamente segnalati. Ogni singolo esperimento, con dei parametri ben fissati, ha avuto una durata complessiva di circa 30 minuti ed ha previsto la percorrenza ripetuta da parte dei veicoli del breve

tratto stradale racchiuso tra le due rotonde a velocità prefissate incrementali. Ogni tratta (da rotonda a rotonda) è stata percorsa quattro volte a *50 Km/h*, *60 Km/h*, *70 Km/h*, *80 Km/h* e *90 Km/h*. Gli esperimenti verranno divisi in gruppi ognuno dei quali con uno specifico scopo e verranno confrontati i loro risultati dando particolare attenzione a parametri come il jitter medio, throughput medio ed il rapporto di invio medio. Verranno inoltre confrontati grafici relativi a parametri di differenti esperimenti per mettere in risalto le differenze nell'arco di tutto l'esperimento. Va infine notato che in tutti questi esperimenti la lettura dei dati *GPS* è sempre stata attiva, mentre il beaconing è stato utilizzato solo negli ultimi esperimenti.
I vari esperimenti verranno confrontati tra loro tramite tabelle e grafici. Per quanto riguarda le tabelle, per ogni esperimento verranno presentate cinque tabelle che confronteranno valori relativi a jitter, throughput, statistiche dei pacchetti, distanza e velocità. Per quanto riguarda i grafici, per ogni esperimento verranno presentati tre grafici che confronteranno l'andamento dell'intera simulazione del jitter, del throughput del dispositivo ricevente ed del rapporto d'invio dei pacchetti. Facciamo subito alcune considerazioni sui tre grafici che verranno proposti di continuo:

- grafico relativo al jitter: sulle ascisse abbiamo i valori incrementali degli intervalli indicanti il passare del tempo, mentre sulle ordinate abbiamo valori che vanno da *0* a *5*. I valori sulle ordinate sono espressi in millisecondi e sono stati tagliati al valore *5* perché sopra a quel valore c'erano solo pochi picchi non rilevanti che avrebbero impedito la visione dell'andamento medio del *jitter*. Va notato che i valori medi sono molto bassi, tipicamente sotto il valore *1*, però il valore *0* indica che non ci sono stati invii e quindi l'impossibilità di calcolare questo parametro durante tali intervalli. Un grafico meno denso con meno picchi è preferibile in quanto vogliamo tenere il valore del *jitter* più basso possibile,

- grafico relativo al throghuput: sulle ascisse abbiamo i valori incrementali degli intervalli indicanti il passare del tempo, mentre sulle ordinate abbiamo valori che vanno da *0* a *64000*. I valori sulle ordinate sono espressi in *bit/s* ed il valore *64000* indica la massima frequenza trasmissiva raggiungibile (parametro fissato in fase di simulazione). Più il grafico è denso (*throughput* alto) e più la banda viene sfruttata, quindi sono preferibili questi tipi di grafico,

- grafico relativo alle statistiche sui pacchetti: sulle ascisse abbiamo i valori incrementali degli intervalli indicanti il passare del tempo, mentre sulle ordinate abbiamo valori che vanno da *0* ad 1.

Il valore *0* in un preciso punto indica che in quell'intervallo, quindi nell'arco di quel secondo, nessuno dei pacchetti inviati è stato ricevuto oppure non è stato inviato alcun pacchetto (periodo di silenzio), invece il valore *1* indica che tutti i pacchetti inviati sono stati ricevuti correttamente. Quindi un grafico molto fitto è preferibile ad uno poco denso.

4.2.1 La lista degli esperimenti

Poiché nel seguito faremo riferimento a dei ben precisi esperimenti, riportiamo nella tabella 4.1 l'elenco di tutti gli esperimenti menzionati in questa tesi con i loro parametri.

Num.	Canale	Data-rate	Size	Throughput	Posizione	Beaconing
1	Controllo	3Mbit/s	80 byte	64000 bit/s	Standard	Inattivo
2	Controllo	6Mbit/s	80 byte	64000 bit/s	Standard	Inattivo
3	Servizio	3Mbit/s	80 byte	64000 bit/s	Standard	Inattivo
4	Controllo	3Mbit/s	160 byte	64000 bit/s	Standard	Inattivo
5	Controllo	3Mbit/s	320 byte	64000 bit/s	Standard	Inattivo
6	Controllo	3Mbit/s	640 byte	64000 bit/s	Standard	Inattivo
7	Controllo	3Mbit/s	80 byte	64000 bit/s	Invertita	Inattivo
8	Controllo	3Mbit/s	80 byte	64000 bit/s	Standard	Inattivo
9	Controllo	3Mbit/s	80 byte	64000 bit/s	Invertita	Inattivo
10	Controllo	3Mbit/s	80 byte	64000 bit/s	Standard	Attivo
11	Controllo	3Mbit/s	160 byte	64000 bit/s	Standard	Attivo
12	Controllo	3Mbit/s	80 byte	16000 bit/s	Standard	Inattivo

Tabella 4.1: Lista dei parametri usati per gli esperimenti.

Prima di iniziare a presentare ed analizzare i risultati ottenuti dobbiamo premettere che alcuni valori, essendo calcolati tramite una discretizzazione al secondo, possono apparire non corretti. I tipici parametri che soffrono di questo problema sono quelli relativi al throughput e ai vari rapporti. Un possibile esempio di questo problema potrebbe essere avere il throughput di un esperimento maggiore rispetto a quello di un secondo esperimento ma al contempo il numero di pacchetti inviati nel primo esperimento potrebbe risultare inferiore rispetto al numero dei pacchetti inviati nel secondo.

4.2.2 La migliore configurazione delle *LinkBird-MX*

I primi esperimenti sono serviti per capire quale configurazione delle *LinkBird-MX* fosse la migliore concentrandosi su due parametri:

- il canale: la scelta è stata effettuata tra il *canale di controllo* ed il *canale di servizio*,

- la velocità trasmissiva: la scelta è stata effettuata tra *3 Mbit/s* e *6 Mbit/s*.

Per questo sono stati svolti tre esperimenti allo scopo di capire quale accoppiamento dei due parametri fosse il migliore. Dal lato applicativo gli esperimenti sono stati effettuati con gli stessi parametri, e cioè ad una frequenza trasmissiva di *64000 bit/s* con pacchetti di *80 byte* ciascuno. Più dettagliatamente i tre esperimenti svolti sono stati:

- primo esperimento sul *canale di controllo a 3 Mbit/s*,
- secondo esperimento sul *canale di controllo a 6 Mbit/s*,
- terzo esperimento sul *canale di servizio a 3 Mbit/s*,

Non è stato necessario utilizzare la configurazione sul canale di servizio a *6 Mbit/s* perché i risultati erano già palesi dopo i primi tre esperimenti. Procederemo confrontando il primo esperimento con il secondo per capire a quale velocità trasmissiva si comporta meglio il sistema sul canale di controllo. La migliore configurazione tra le due verrà poi confrontata con quella del terzo esperimento per capire se ci conviene trasmettere sul canale di controllo o sul canale di servizio. Questi primi tre esperimenti sono stati svolti leggendo i dati del dispositivo *GPS* e non sono stati influenzati dal beaconing.

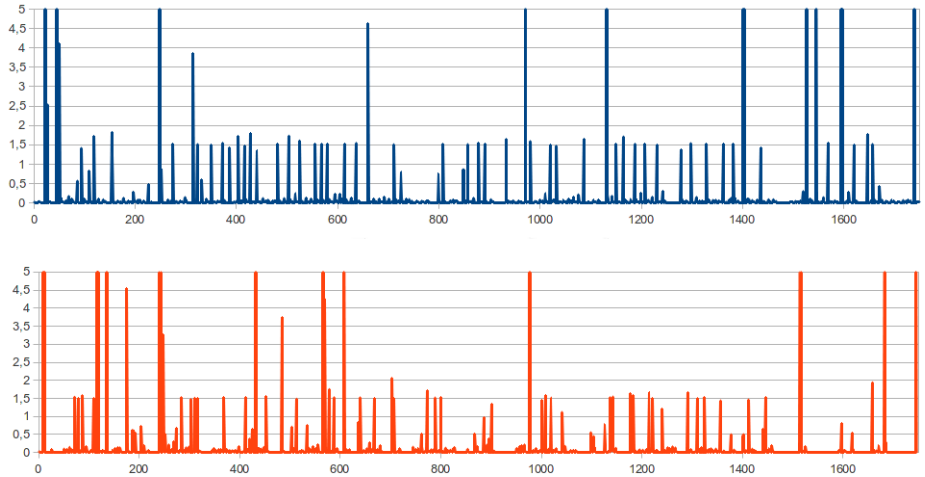

Figura 4.2: Andamento del *Jitter* del primo e del secondo esperimento, le ascisse sono espresse in secondi mentre le ordinate sono espresse in millisecondi.

Cominciamo dunque con il primo confronto riportando i valori medi dei parametri nelle tabelle. I risultati sono già eloquenti, ma riportiamo per completezza alcuni grafici per confrontare l'andamento di alcuni parametri durante l'intero esperimento. Il primo parametro preso in considerazione è il *Jitter* di cui riportiamo i valori medi nella tabella 4.2 e l'andamento durante tutto l'esperimento

	Primo Esperimento	Secondo Esperimento
Jitter Minimo	2.00103609250310E-9	4.33170507928173E-6
Jitter Massimo	424.7175441883311	381.0781017081293
Jitter Medio	0.15502994465923736	0.2027442371020856

Tabella 4.2: Valori medi del *Jitter* del primo e del secondo esperimento espressi in millisecondi.

nella figura 4.2. Notiamo subito che il *jitter* minimo del primo esperimento è di gran lunga più piccolo di quello del secondo (E^{-9} contro E^{-6}) mentre il picco massimo premia il secondo esperimento (circa 424 millisecondi contro 381), quello che importa però è il *jitter* medio che premia, seppur di poco, il primo esperimento (circa 0,15 millisecondi contro circa 0,20 millisecondi). Entrambi gli esperimenti hanno valori medi molto buoni dato che le comunicazioni *VoIP* possono sostenere valori di *jitter* anche di 300 millisecondi. Per quanto riguarda l'andamento dell'intero esperimento riportato nella figura 4.2, i due grafici riportati sono molto simili nell'andamento medio, ma si può notare come il secondo grafico abbia un maggior numero di picchi e sia più denso soprattutto nei valori molto piccoli (sotto *0,5*).

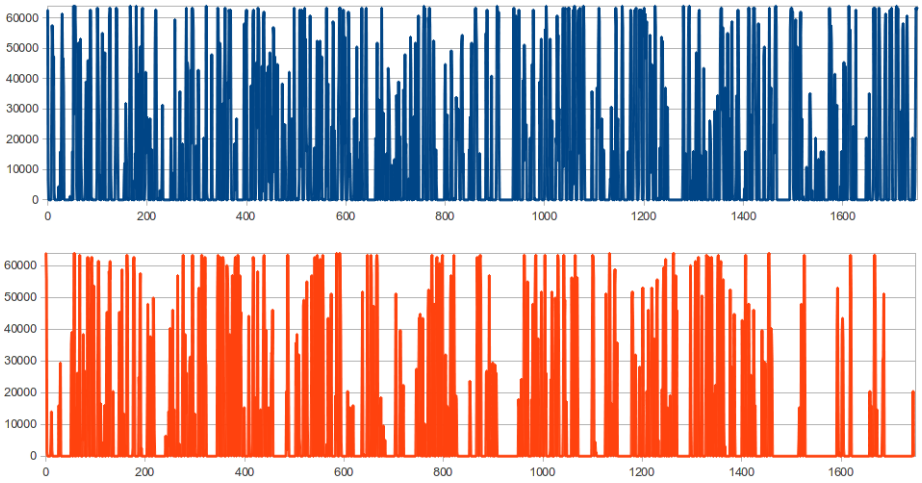

Figura 4.3: Andamento del *Throughput* del dispositivo ricevente del primo e del secondo esperimento, le ascisse sono espresse in secondi mentre le ordinate sono espresse in bit/s.

Proseguiamo prendendo in considerazione il throughput riportando i valori medi di entrambi i dispositivi nella tabella 4.3 e l'andamento dell'intera simulazione del throughput del dispositivo ricevente nella figura 4.3. Vediamo subito come i valori minimi e massimo del throughput siano poco interessanti mentre è molto utile analizzare i valori medi. Il throughput medio del dispositivo inviante

	Primo Esperimento	Secondo Esperimento
Throughput Minimo (I)	640	640
Throughput Massimo (I)	64000	64000
Throughput Medio (I)	40176.19210977702	42183.90885188431
Throughput Minimo (R)	0	0
Throughput Massimo (R)	64000	64000
Throughput Medio (R)	21616.7409948542	15293.812445223488
Rapporto Throughput	0.5380485272419258	0.36255086030369915

Tabella 4.3: Valori medi del *Throughput* del primo e del secondo esperimento espressi in bit/s.

del primo esperimento è inferiore a quello del secondo esperimento (circa 40176 bit/s conto circa 42183 bit/s), questo succede perché il throughput del dispositivo inviante viene soprattutto influenzato dai periodi di silenzio. Nonostante un valore medio di throughput nel dispositivo di invio maggiore nel secondo esperimento corrisponde un valore di throughput del dispositivo ricevente inferiore nel secondo esperimento nei confronti del primo esperimento (circa 21616 bit/s contro circa 15293 bit/s). Questo risultato viene sottolineato ancor più dal rapporto tra i throughput che tiene in considerazione il throughput d'invio e quello di ricezione dei dispositivi (circa 0,53 del primo esperimento contro circa 0,36 del secondo esperimento). Per quanto riguarda l'andamento dell'intera simulazione mostrato nella figura 4.3, si può subito notare come il primo grafico sia più denso rispetto al secondo, soprattutto nella fase finale, e di conseguenza possiamo concludere che nel primo esperimento il *throughput* si comporta meglio rispetto al secondo.

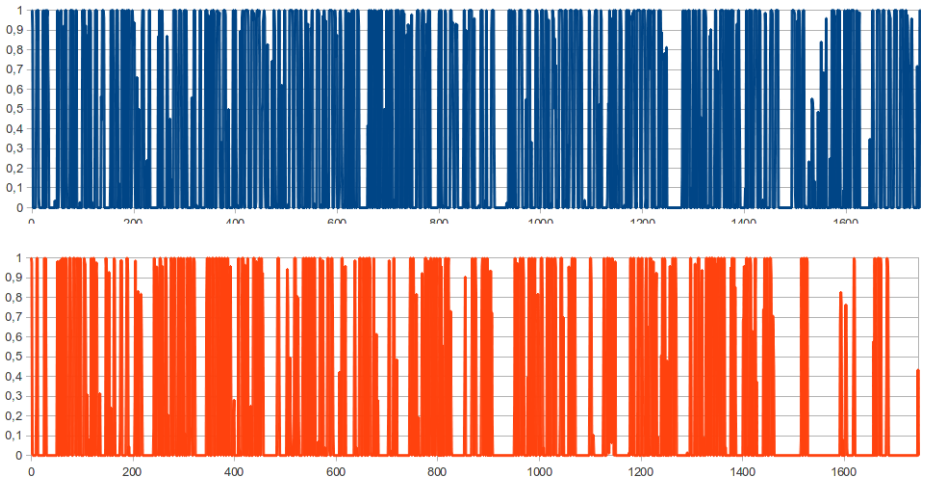

Figura 4.4: Andamento del rapporto d'invio del primo e del secondo esperimento, le ascisse sono espresse in secondi mentre le ordinate vanno da 0 ad 1.

	Primo Esperimento	Secondo Esperimento
Pacchetti Inviati	73506	75499
Pacchetti Ricevuti	39513	27303
Pacchetti Persi	33993	48196
Rapporto Medio	0.5375479552689576	0.36163392892621093
Numero intervalli vuoti	391	571
Massimo intervallo vuoto	17	31

Tabella 4.4: Statistiche dei pacchetti del primo e del secondo esperimento.

Vediamo ora le statistiche riguardanti gli invii e le ricezioni dei pacchetti nella tabella 4.4 e l'andamento del rapporto d'invio nella figura 4.4. Sono stati inviati più pacchetti nel secondo esperimento (73506 contro 75499) però sono stati ricevuti correttamente più pacchetti nel primo esperimento (39513 contro 27303) e, di conseguenza, le perdite sono state maggiori nel secondo esperimento (33993 contro 48196). Infatti il rapporto di invio risulta maggiore nel primo esperimento (circa 0,53 contro circa 0,36). Coerentemente con queste statistiche il numero di intervalli in cui non sono stati ricevuti nessuno dei pacchetti inviati sono maggiori nel secondo esperimento (391 contro 571) e l'intervallo massimo vuoto risulta maggiore nel secondo esperimento (17 contro 31). Per quanto riguarda l'andamento dell'intera simulazione mostrato nella figura 4.4, essendo il primo grafico più denso rispetto al secondo, soprattutto nella parte finale, possiamo concludere che l'andamento medio del primo grafico risulta migliore di quello del secondo.

	Primo Esperimento	Secondo Esperimento
Distanza Minima	1.7077971554148954	0.18899329649861554
Distanza Massima	105.08872826648025	76.11267834891504
Distanza Media	25.992020109110552	24.233411265650854

Tabella 4.5: Valori medi delle distanze del primo e del secondo esperimento espressi in metri.

	Primo Esperimento	Secondo Esperimento
Velocità Minima (I)	0	0
Velocità Massima (I)	24.93	24.15
Velocità Media (I)	15.05332667913325	14.499261278402265
Velocità Minima (R)	0	0
Velocità Massima (R)	28.5	23.74
Velocità Media (R)	14.965321675847127	14.5099257253667

Tabella 4.6: Valori medi della velocità del primo e del secondo esperimento espressi in metri al secondo.

Riportiamo infine i valori medi di distanza e velocità nelle tabelle 4.5 e 4.6.

Dopo tutte queste analisi possiamo concludere che, nonostante valori medi simili di distanza e velocità, le perdite sono molto più alte nel secondo caso, il *throughput* del dispositivo ricevente è mediamente maggiore nel primo caso ed il *jitter*, escludendo alcuni picchi isolati, è mediamente inferiore nel primo caso. Dunque possiamo concludere che il canale di controllo si comporta meglio utilizzando una velocità trasmissiva di *3 Mbit/s* piuttosto che *6 Mbit/s*.

Andiamo quindi ora a confrontare il primo esperimento con il terzo per capire se sia meglio sfruttare il canale di controllo o quello di servizio.

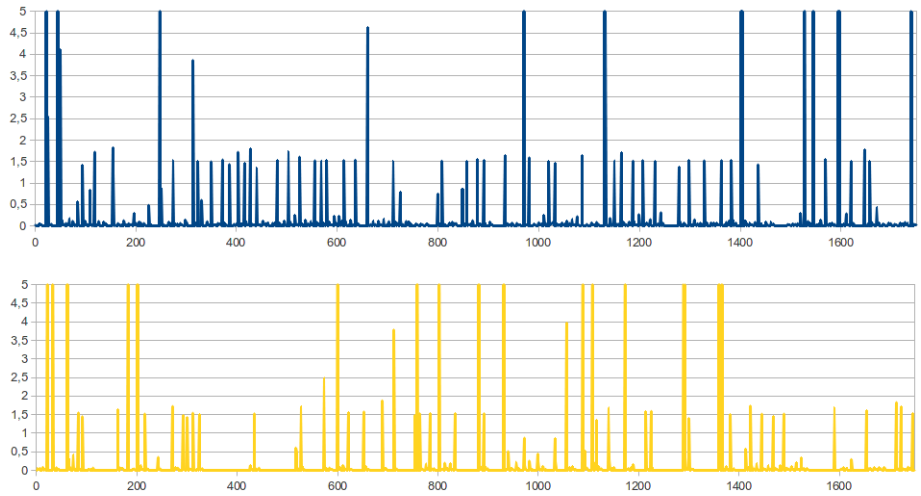

Figura 4.5: Andamento del *jitter* del primo e del terzo esperimento, le ascisse sono espresse in secondi mentre le ordinate sono espresse in millisecondi.

	Primo Esperimento	Terzo Esperimento
Jitter Minimo	2.0010360925031E-9	4.48505950309069E-9
Jitter Massimo	424.7175441883311	464.319887190864
Jitter Medio	0.15502994465923736	0.23283727342767932

Tabella 4.7: Valori medi del *Jitter* del primo e del terzo esperimento espressi in millisecondi.

Cominciamo questo secondo confronto tra il primo esperimento ed il terzo analizzando il *Jitter* di cui riportiamo i valori medi nella tabella 4.7 e l'andamento durante tutta la simulazione nella figura 4.5. Notiamo subito che il *jitter* minimo e quello massimo del primo esperimento e del terzo esperimento sono sostanzialmente simili (entrambi ordine E^{-9} per il minimo, circa 424 millisecondi

contro 464 millisecondi per il massimo), il *jitter* medio però premia ancora una volta, ancor più rispetto al confronto precedente, il primo esperimento (circa 0,15 millisecondi contro circa 0,23 millisecondi). Anche il jitter medio del terzo esperimento risulta accettabile per le comunicazioni *VoIP*. Per quanto riguarda l'andamento dell'intera simulazione riportato nella figura 4.5, i due grafici riportati sono molto simili nell'andamento medio, ma si può notare come il secondo grafico abbia un maggior numero di picchi, sia più denso soprattutto nei valori molto piccoli (sotto *0,5*) ed inoltre abbia molti più periodi in cui non è stato possibile calcolare il *jitter*.

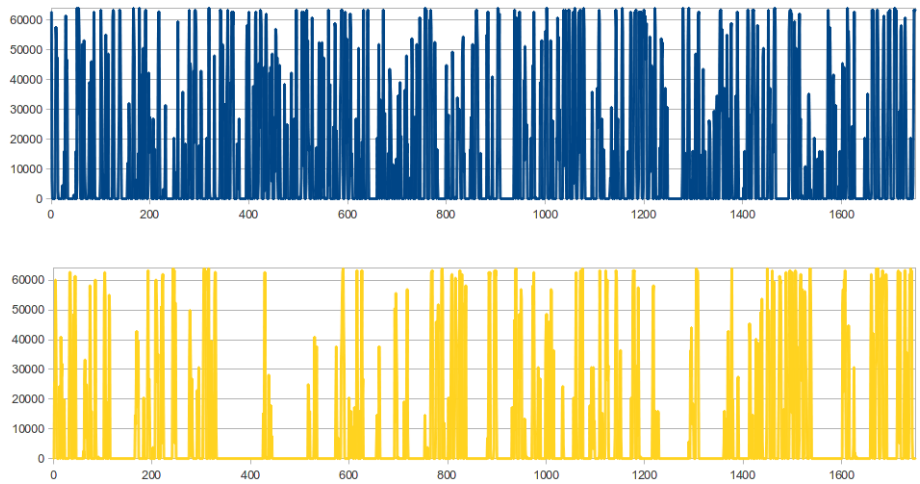

Figura 4.6: Andamento del *Throughput* del dispositivo ricevente del primo e del terzo esperimento, le ascisse sono espresse in secondi mentre le ordinate sono espresse in bit/s.

	Primo Esperimento	Terzo Esperimento
Throughput Minimo (I)	640	640
Throughput Massimo (I)	64000	64000
Throughput Medio (I)	40176.19210977702	40552.151215121514
Throughput Minimo (R)	0	0
Throughput Massimo (R)	64000	64000
Throughput Medio (R)	21616.7409948542	14122.628262826283
Rapporto Throughput	0.5380485272419258	0.34825842377407806

Tabella 4.8: Valori medi del *Throughput* del primo e del terzo esperimento espressi in bit/s.

Proseguiamo prendendo in considerazione il throughput del secondo confronto riportando i valori medi di entrambi i dispositivi nella tabella 4.8 e l'andamento dell'intera simulazione del throughput del dispositivo ricevente nella figura 4.6. Ancora una volta i valori minimi e massimo del throughput sono poco

interessanti mentre è molto utile analizzare i valori medi. Il throughput medio del dispositivo inviante del primo esperimento è inferiore di pochissimo a quello del terzo esperimento (circa 40176 bit/s conto circa 40552 bit/s), facendoci intuire come i periodi di silenzio e parlato totali di questi due esperimenti siano molto simili. Per quanto riguarda il throughput medio del dispositivo ricevente, quello del terzo esperimento risulta di poco inferiore a quello del secondo esperimento e quindi maggiormente inferiore a quello del primo esperimento (circa 21616 bit/s contro circa 14122 bit/s). Questo risultato viene sottolineato ancor più dal rapporto tra i throughput (circa 0,53 del primo esperimento contro circa 0,34 del terzo esperimento). Per quanto riguarda l'andamento dell'intera simulazione mostrato nella figura 4.6, si può subito notare come il primo grafico sia più denso rispetto al secondo, soprattutto in alcuni punti sparsi in cui nel secondo grafico ci sono dei 'buchi' vistosi in cui il throughput va a zero, e di conseguenza possiamo concludere che nel primo esperimento il *throughput* si comporta meglio rispetto al terzo esperimento.

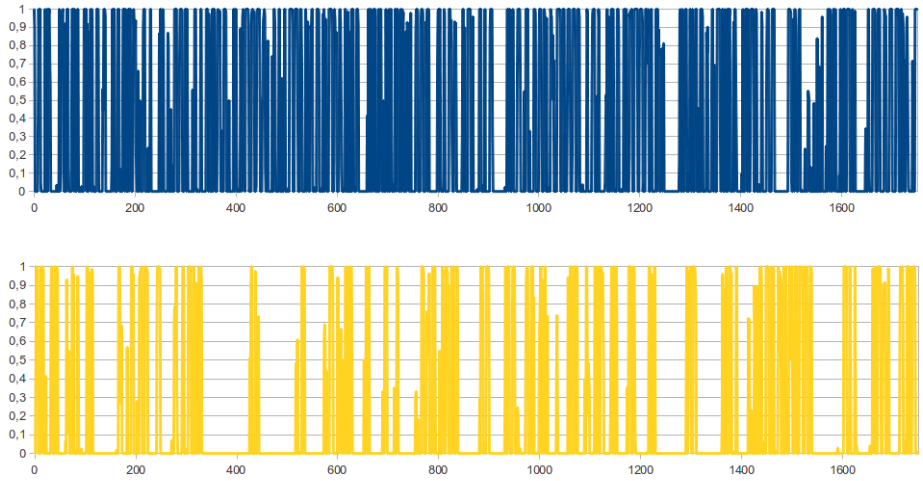

Figura 4.7: Andamento del rapporto d'invio del primo e del terzo esperimento, le ascisse sono espresse in secondi mentre le ordinate vanno da 0 a 1.

	Primo Esperimento	Terzo Esperimento
Pacchetti Inviati	73506	70669
Pacchetti Ricevuti	39513	24621
Pacchetti Persi	33993	46048
Rapporto Medio	0.5375479552689576	0.34839887362209737
Numero intervalli vuoti	391	626
Massimo intervallo vuoto	17	13

Tabella 4.9: Statistiche dei pacchetti del primo e del terzo esperimento.

Vediamo ora le statistiche rigardanti gli invii e le ricezioni dei pacchetti nella tabella 4.9 e l'andamento del rapporto d'invio nella figura 4.7. Sono stati inviati più pacchetti nel primo esperimento (73506 contro 70669) e sono stati ricevuti correttamente molti più pacchetti nel primo esperimento (39513 contro 24621) e, di conseguenza, le perdite sono state maggiori nel terzo esperimento (33993 contro 46048). Il rapporto di invio risulta di conseguenza maggiore nel primo esperimento (circa 0,53 contro circa 0,34). Coerentemente con queste statistiche il numero di intervalli in cui non sono stati ricevuti nessuno dei pacchetti inviati sono maggiori nel terzo esperimento (391 contro 626) però stavolta l'intervallo massimo vuoto risulta maggiore nel primo esperimento (17 contro 13). Per quanto riguarda l'andamento dell'intera simulazione mostrato nella figura 4.7, sulle ascisse abbiamo i valori incrementali degli intervalli indicanti il passare del tempo, mentre sulle ordinate abbiamo valori che vanno da *0* ad 1.
Il valore *0* in un preciso punto indica che in quell'intervallo, quindi nell'arco di quel secondo, nessuno dei pacchetti inviati è stato ricevuto oppure non è stato inviato alcun pacchetto (periodo di silenzio), invece il valore *1* indica che tutti i pacchetti inviati sono stati ricevuti correttamente. Quindi un grafico molto fitto è preferibile ad uno poco denso. Essendo il primo grafico più denso rispetto al secondo, soprattutto in alcuni punti distribuiti nell'arco di tutta la simulazione, possiamo concludere che il primo grafico ha un andamento migliore rispetto al secondo.

	Primo Esperimento	Terzo Esperimento
Distanza Minima	1.7077971554148954	0.8541903575988428
Distanza Massima	105.08872826648025	64.72196923567364
Distanza Media	25.992020109110552	21.72248608387508

Tabella 4.10: Valori medi delle distanze del primo e del terzo esperimento espressi in metri.

	Primo Esperimento	Terzo Esperimento
Velocità Minima (I)	0	0
Velocità Massima (I)	24.93	24.86
Velocità Media (I)	15.05332667913325	14.430578622982539
Velocità Minima (R)	0	0
Velocità Massima (R)	28.5	26.025
Velocità Media (R)	14.965321675847127	14.289362744303753

Tabella 4.11: Valori medi della velocità del primo e del terzo esperimento espressi in metri al secondo.

Riportiamo infine i valori medi di distanza e velocità nelle tabelle 4.10 e 4.11.

Quindi, dopo queste analisi, possiamo concludere che i confronti dei valori medi e dei grafici del primo esperimento con quelli del terzo esperimento danno lo stesso risultato del precedente confronto ed addirittura le differenze risultano maggiori. Quindi, dopo tutti questi confronti, possiamo concludere che la configurazione delle *LinkBird-MX* che prevede l'uso del canale di controllo a *3 Mbit/s* risulta la migliore possibile.

4.2.3 La miglior dimensione dei pacchetti

Adesso che abbiamo capito quale sia la miglior configurazione da usare per le *LinkBird-MX* (*3 Mbit/s* sul canale di controllo), ci possiamo concentrare sulla dimensione ottimale dei pacchetti inviati, scegliendo tra:

- 80 byte,
- 160 byte,
- 320 byte,
- 640 byte.

Tutti e quattro gli esperimenti sono stati svolti con la configurazione migliore descritta nel paragrafo precedente e disabilitando il beaconing. Cominciamo col confrontare l'esperimento ad 80 byte con quello a 160 byte.

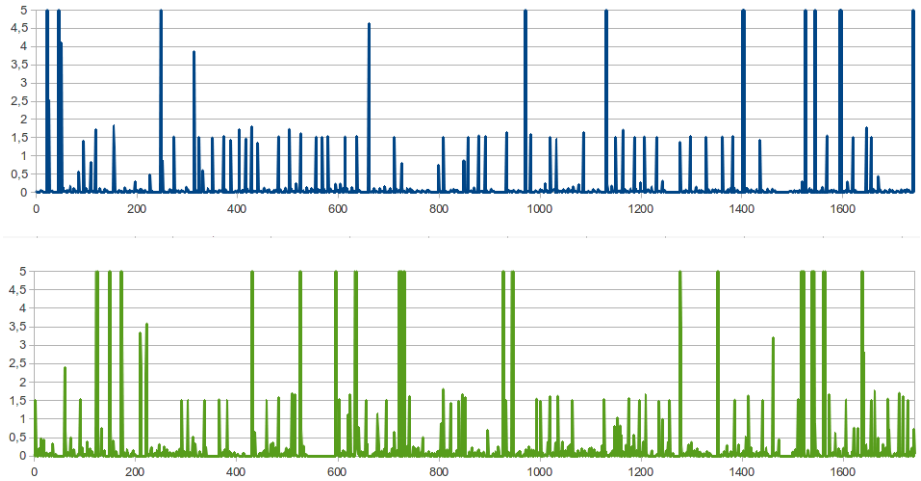

Figura 4.8: Andamento del *jitter* del primo e del quarto esperimento, le ascisse sono espresse in secondi mentre le ordinate sono espresse in millisecondi.

Cominciamo il confronto tra 80 byte e 160 byte analizzando il *Jitter* di cui riportiamo i valori medi nella tabella 4.12 e l'andamento durante tutta la

	80 byte	160 byte
Jitter Minimo	2.00103609250310E-9	6.72429392825798E-8
Jitter Massimo	424.7175441883311	437.1474631242738
Jitter Medio	0.15502994465923736	0.29301558752997603

Tabella 4.12: Valori medi del *Jitter* del primo e del quarto esperimento espressi in millisecondi.

simulazione nella figura 4.8. Il *jitter* minimo del primo esperimento risulta inferiore di un'ordine di grandezza rispetto al quarto (ordine E^{-9} contro ordine E^{-8}) mentre quello massimo dei due esperimenti risulta pressoché identico (circa 424 millisecondi contro circa 437 millisecondi). Il *jitter* medio però premia il primo esperimento (circa 0,15 millisecondi contro circa 0,29 millisecondi). Per quanto riguarda l'andamento dell'intera simulazione riportato nella figura 4.8, i due grafici riportati sono molto simili nell'andamento medio, ma si può notare come il secondo grafico abbia un maggior numero di picchi, sia più denso soprattutto nei valori molto piccoli (sotto *0,5*) ed abbia un vistoso periodo in cui non è stato possibile calcolare il *jitter* immediatamente prima del valore 600.

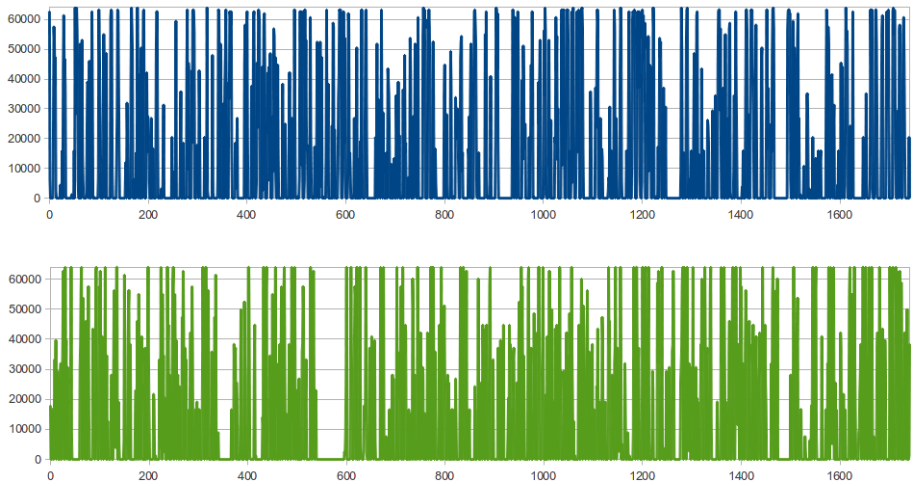

Figura 4.9: Andamento del *Throughput* del dispositivo ricevente del primo e del quarto esperimento, le ascisse sono espresse in secondi mentre le ordinate sono espresse in bit/s.

Proseguiamo il confronto prendendo in considerazione il throughput riportando i valori medi di entrambi i dispositivi nella tabella 4.13 e l'andamento dell'intera simulazione del throughput del dispositivo ricevente nella figura 4.9. Ancora una volta i valori minimi e massimo del throughput sono poco interessanti mentre è molto utile analizzare i valori medi. Il throughput medio del dispositivo inviante del primo esperimento è di poco superiore a quello del

	80 byte	160 byte
Throughput Minimo (I)	640	1280
Throughput Massimo (I)	64000	64000
Throughput Medio (I)	40176.19210977702	39381.70434782609
Throughput Minimo (R)	0	0
Throughput Massimo (R)	64000	64000
Throughput Medio (R)	21616.7409948542	21054.33043478261
Rapporto Throughput	0.5380485272419258	0.5346221242439658

Tabella 4.13: Valori medi del *Throughput* del primo e del quarto esperimento espressi in bit/s.

quarto esperimento (circa 40176 bit/s conto circa 39381 bit/s), facendoci intuire come i periodi di silenzio e parlato totali di questi due esperimenti siano molto simili. Per quanto riguarda il throughput medio del dispositivo ricevente, quello del quarto esperimento risulta di poco inferiore a quello del primo esperimento (circa 21616 bit/s contro circa 21054 bit/s). Questo risultato viene sottolineato ancor più dal rapporto tra i throughput quasi identico (circa 0,538 del primo esperimento contro circa 0,534 del quarto esperimento). Per quanto riguarda l'andamento dell'intera simulazione mostrato nella figura 4.9, i due grafici sono molto diversi e quindi difficilemente comparabili, unica nota è il vuoto nel quarto esperimento prima del valore 600. Possiamo dunque concludere che i due comportamenti sono molto simili pur avendo andamento diverso.

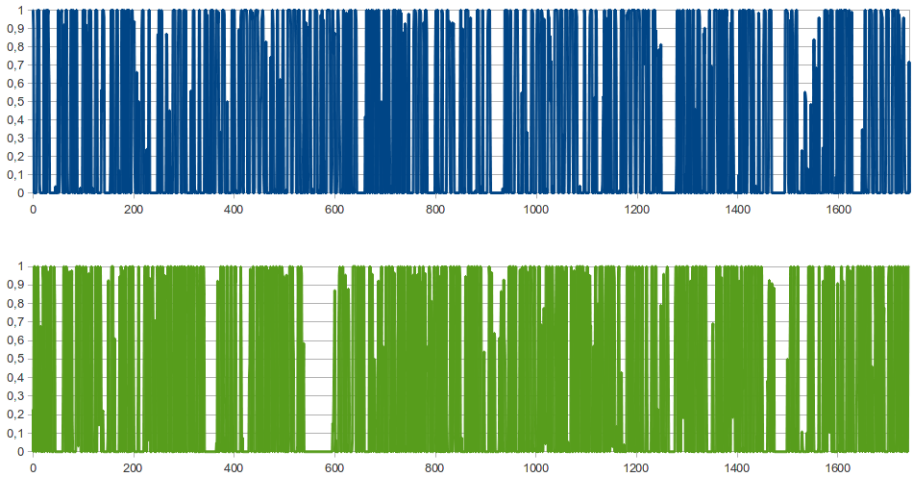

Figura 4.10: Andamento del rapporto d'invio del primo e del quarto esperimento, le ascisse sono espresse in secondi mentre le ordinate vanno da 0 a 1.

Vediamo ora le statistiche rigardanti gli invii e le ricezioni dei pacchetti nella tabella 4.14 e l'andamento del rapporto d'invio nella figura 4.10. Data

	80 byte	160 byte
Pacchetti Inviati	73506	38556
Pacchetti Ricevuti	39513	19967
Pacchetti Persi	33993	18589
Rapporto Medio	0.5375479552689576	0.5178701110073659
Numero intervalli vuoti	391	329
Massimo intervallo vuoto	17	15

Tabella 4.14: Statistiche dei pacchetti del primo e del quarto esperimento.

la differenza nella dimensione dei pacchetti, sono stati inviati circa il doppio dei pacchetti nel primo esperimento rispetto al quarto (73506 contro 38556) ed ovviamente sono stati ricevuti correttamente molti più pacchetti nel primo esperimento (39513 contro 19967) e le perdite sono state maggiori nel primo esperimento (33993 contro 18589). In questo caso dobbiamo affidarci al rapporto di invio per capire meglio i risultati, e questo premia di poco il primo esperimento (circa 0,53 contro circa 0,51). Invece, per quanto riguarda lo studio delle perdite, il numero di intervalli in cui non sono stati ricevuti nessuno dei pacchetti inviati sono maggiori nel primo esperimento (391 contro 329) e l'intervallo massimo vuoto risulta maggiore nel primo esperimento (17 contro 15). Per quanto riguarda l'andamento dell'intera simulazione mostrato nella figura 4.10, il secondo grafico risulta mediamente più denso rispetto al primo, ma soffre di vuoti maggiori soprattutto in alcuni punti distribuiti nell'arco di tutta la simulazione. Possiamo quindi concludere che i due grafici hanno un andamento molto simile.

	80 byte	160 byte
Distanza Minima	1.7077971554148954	0.4848244257621118
Distanza Massima	105.08872826648025	92.95214940084924
Distanza Media	25.992020109110552	23.421475432946607

Tabella 4.15: Valori medi delle distanze del primo e del quarto esperimento espressi in metri.

	80 byte	160 byte
Velocità Minima (I)	0	0
Velocità Massima (I)	24.93	24.979
Velocità Media (I)	15.05332667913325	15.32631058600146
Velocità Minima (R)	0	0
Velocità Massima (R)	28.5	26.56
Velocità Media (R)	14.965321675847127	15.261059099765257

Tabella 4.16: Valori medi della velocità del primo e del quarto esperimento espressi in metri al secondo.

Riportiamo infine i valori medi della distanza e della velocità nelle tabelle 4.15 e 4.16.

Dopo queste statistiche possiamo concludere che la dimensione dei pacchetti ad 80 byte e a 160 byte portano ad esperimenti con risultati quasi uguali premiando di pochissimo la dimensione a 80 byte. Contrariamente al confronto precedente molto equilibrato, i due successivi confronti sono molto squilibrati.

Confrontiamo ora l'esperimento con pacchetti a 80 byte con quello con pacchetti da 320 byte.

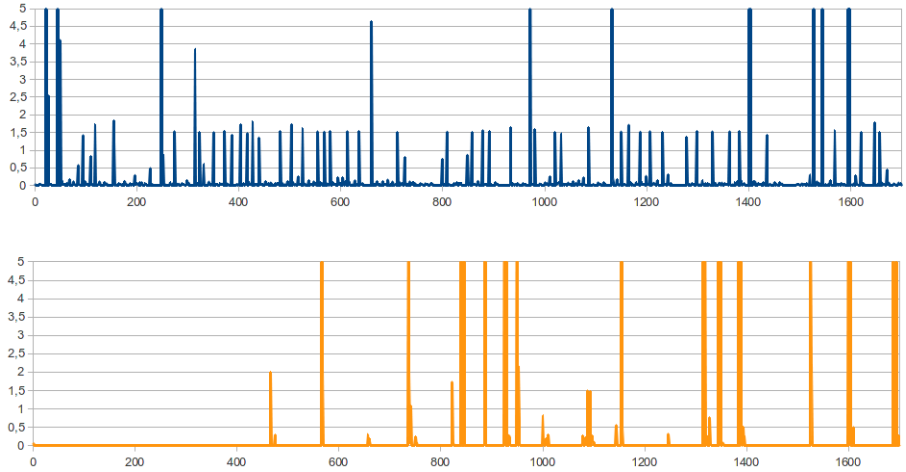

Figura 4.11: Andamento del *jitter* del primo e del quinto esperimento, le ascisse sono espresse in secondi mentre le ordinate sono espresse in millisecondi.

	80 byte	320 byte
Jitter Minimo	2.00103609250310E-9	3.49396811595103E-5
Jitter Massimo	424.7175441883311	465.89551614880423
Jitter Medio	0.15502994465923736	5.284114052953157

Tabella 4.17: Valori medi del *Jitter* del primo e del quinto esperimento espressi in millisecondi.

Cominciamo confrontando il *Jitter* di cui riportiamo i valori medi nella tabella 4.17 e l'andamento durante tutta la simulazione nella figura 4.11. Il *jitter* minimo del primo esperimento risulta inferiore di quattro ordini di grandezza rispetto al quinto (ordine E^{-9} contro ordine E^{-5}) mentre quello massimo dei due esperimenti risulta molto vicino (circa 424 millisecondi contro circa 465 millisecondi). Il *jitter* medio premia di gran lunga il primo esperimento (circa 0,15

millisecondi contro circa 5 millisecondi) e per la prima volta abbiamo un jitter medio superiore al millisecondo. Per quanto riguarda l'andamento dell'intera simulazione riportato nella figura 4.11, i due grafici riportati sono molto diversi in quanto il primo ha alcuni picchi e per il resto ha un andamento costante su valori piccoli mentre il secondo alterna picchi (molti più del primo) a valori nulli a causa delle enormi perdite di pacchetti con pochi valori intermedi. Quindi è decisamente preferibile l'andamento del primo grafico.

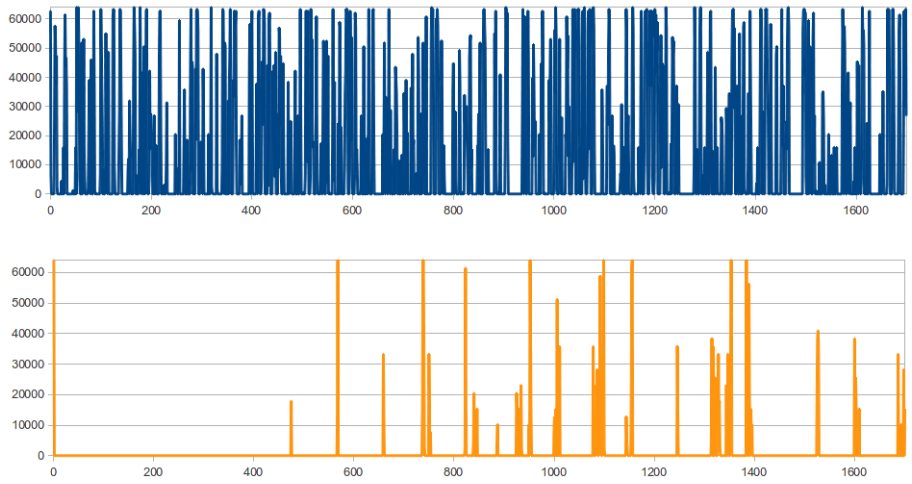

Figura 4.12: Andamento del *Throughput* del dispositivo ricevente del primo e del quinto esperimento, le ascisse sono espresse in secondi mentre le ordinate sono espresse in bit/s.

	80 byte	320 byte
Throughput Minimo (I)	640	2560
Throughput Massimo (I)	64000	64000
Throughput Medio (I)	40176.19210977702	39503.76811594203
Throughput Minimo (R)	0	0
Throughput Massimo (R)	64000	64000
Throughput Medio (R)	21616.7409948542	2177.391304347826
Rapporto Throughput	0.5380485272419258	0.05511857243484385

Tabella 4.18: Valori medi del *Throughput* del primo e del quinto esperimento espressi in bit/s.

Proseguiamo il confronto prendendo in considerazione il throughput riportando i valori medi di entrambi i dispositivi nella tabella 4.18 e l'andamento dell'intera simulazione del throughput del dispositivo ricevente nella figura 4.12. Ancora una volta i valori minimi e massimo del throughput sono poco interessanti mentre è molto utile analizzare i valori medi. Il throughput medio del dispositivo inviante del primo esperimento è di poco superiore a quello del

quinto esperimento (circa 40176 bit/s conto circa 39503 bit/s), facendoci intuire come i periodi di silenzio e parlato totali di questi due esperimenti siano molto simili. Per quanto riguarda il throughput medio del dispositivo ricevente, quello del quinto esperimento risulta bassissimo rispetto a quello del primo esperimento (circa 21616 bit/s contro circa 2177 bit/s). Questo risultato viene sottolineato ancor più dal rapporto tra i throughput (circa 0,538 del primo esperimento contro circa 0,05 del quinto esperimento). Per quanto riguarda l'andamento dell'intera simulazione mostrato nella figura 4.12, si nota subito come il secondo grafico sia quasi interamente schiacciato a zero e poco denso soprattutto nella prima parte. Possiamo facilmente concludere che il comportamento del secondo grafico è inaccettabile rispetto al primo.

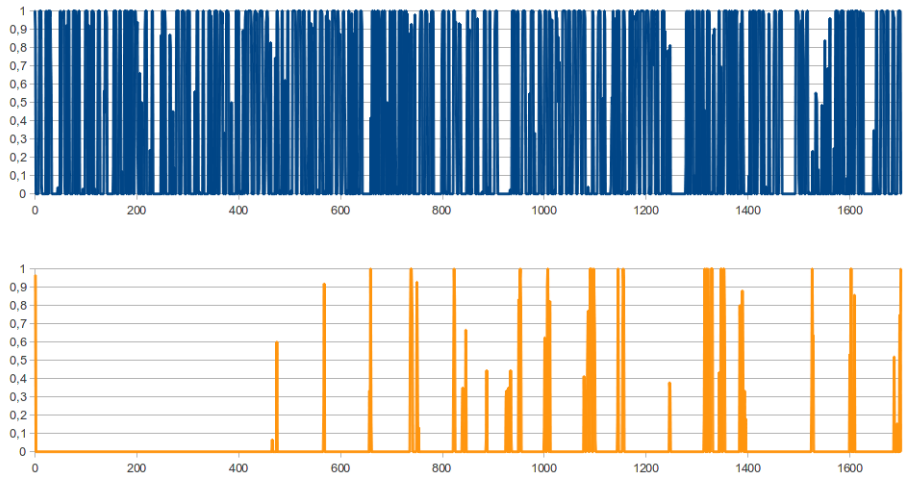

Figura 4.13: Andamento del rapporto d'invio del primo e del quinto esperimento, le ascisse sono espresse in secondi mentre le ordinate vanno da 0 a 1.

	80 byte	320 byte
Pacchetti Inviati	73506	18546
Pacchetti Ricevuti	39513	977
Pacchetti Persi	33993	17569
Rapporto Medio	0.5375479552689576	0.052679823142456596
Numero intervalli vuoti	391	1006
Massimo intervallo vuoto	17	20

Tabella 4.19: Statistiche dei pacchetti del primo e del quinto esperimento.

Vediamo ora le statistiche rigardanti gli invii e le ricezioni dei pacchetti nella tabella 4.19 e l'andamento del rapporto d'invio nella figura 4.13. Data la differenza nella dimensione dei pacchetti, sono stati inviati circa un quarto dei pacchetti nel primo esperimento rispetto al quinto (73506 contro 18546)

ed ovviamente sono stati ricevuti correttamente molti più pacchetti nel primo esperimento (39513 contro 977) e le perdite sono state maggiori nel primo esperimento (33993 contro 17569). In questo caso dobbiamo affidarci al rapporto di invio per capire meglio i risultati, e questo premia di molto il primo esperimento (circa 0,53 contro circa 0,05). Invece, per quanto riguarda lo studio delle perdite, il numero di intervalli in cui non sono stati ricevuti nessuno dei pacchetti inviati sono maggiori nel quinto esperimento (391 contro 1006) e l'intervallo massimo vuoto risulta maggiore sempre nel quinto esperimento seppur di poco (17 contro 20). Per quanto riguarda l'andamento dell'intera simulazione mostrato nella figura 4.13, il secondo grafico risulta poco denso e tendente allo zero a causa delle enormi perdite quindi possiamo concludere che il secondo grafico ha un andamento inaccettabile rispetto al primo.

	80 byte	320 byte
Distanza Minima	1.7077971554148954	1.3061709414724936
Distanza Massima	105.08872826648025	81.25517023611603
Distanza Media	25.992020109110552	14.683073659483762

Tabella 4.20: Valori medi delle distanze del primo e del quinto esperimento espressi in metri.

	80 byte	320 byte
Velocità Minima (I)	0	0
Velocità Massima (I)	24.93	21.83
Velocità Media (I)	15.05332667913325	13.36151340206176
Velocità Minima (R)	0	0
Velocità Massima (R)	28.5	22.06
Velocità Media (R)	14.965321675847127	13.135804123711324

Tabella 4.21: Valori medi della velocità del primo e del quinto esperimento espressi in metri al secondo.

Riportiamo infine i valori medi della distanza e della velocità nelle tabelle 4.20 e 4.21.

Dopo queste statistiche possiamo concludere che la dimensione dei pacchetti ad 80 byte è largamente preferibile a quella a 320 byte a causa delle enormi perdite di quest'ultima.

Concludiamo confrontando la configurazione con 80 byte con quella a 640 byte.

Cominciamo confrontando il *Jitter* di cui riportiamo i valori medi nella tabella 4.22 e l'andamento durante tutta la simulazione nella figura 4.14. Il *jitter* minimo del primo esperimento risulta inferiore di quattro ordini di grandezza ri-

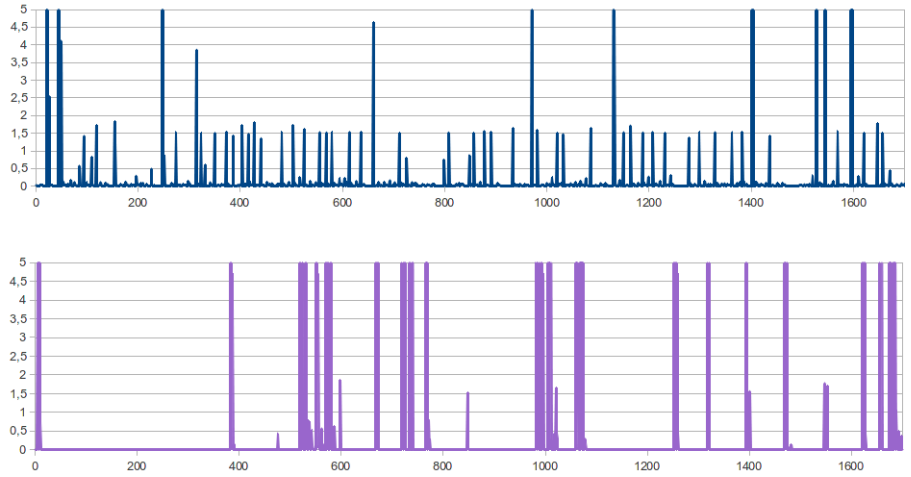

Figura 4.14: Andamento del *jitter* del primo e del sesto esperimento, le ascisse sono espresse in secondi mentre le ordinate sono espresse in millisecondi.

	80 byte	640 byte
Jitter Minimo	2.00103609250310E-9	6.575832723315E-5
Jitter Massimo	424.7175441883311	878.7461908902841
Jitter Medio	0.15502994465923736	13.893848009650181

Tabella 4.22: Valori medi del *Jitter* del primo e del sesto esperimento espressi in millisecondi.

spetto al sesto (ordine E^{-9} contro ordine E^{-5}) mentre quello massimo del sesto esperimento risulta doppio rispetto al primo (circa 424 millisecondi contro circa 878 millisecondi). Il *jitter* medio premia di gran lunga il primo esperimento (circa 0,15 millisecondi contro circa 13 millisecondi) anche questa volta abbiamo un jitter medio superiore al millisecondo. Per quanto riguarda l'andamento dell'intera simulazione riportato nella figura 4.14, i due grafici riportati sono molto diversi in quanto il primo ha alcuni picchi e per il resto ha un andamento costante su valori piccoli mentre il secondo alterna picchi (molti più del primo) a valori nulli a causa delle enormi perdite di pacchetti con pochi valori intermedi. Quindi è decisamente preferibile l'andamento del primo grafico.

Proseguiamo il confronto prendendo in considerazione il throughput riportando i valori medi di entrambi i dispositivi nella tabella 4.23 e l'andamento dell'intera simulazione del throughput del dispositivo ricevente nella figura 4.15. Ancora una volta i valori minimi e massimo del throughput sono poco interessanti mentre è molto utile analizzare i valori medi. Il throughput medio del dispositivo inviante del primo esperimento è di poco superiore a quello del sesto esperimento (circa 40176 bit/s conto circa 38794 bit/s), facendoci intuire come i periodi di silenzio e parlato totali di questi due esperimenti siano molto simili.

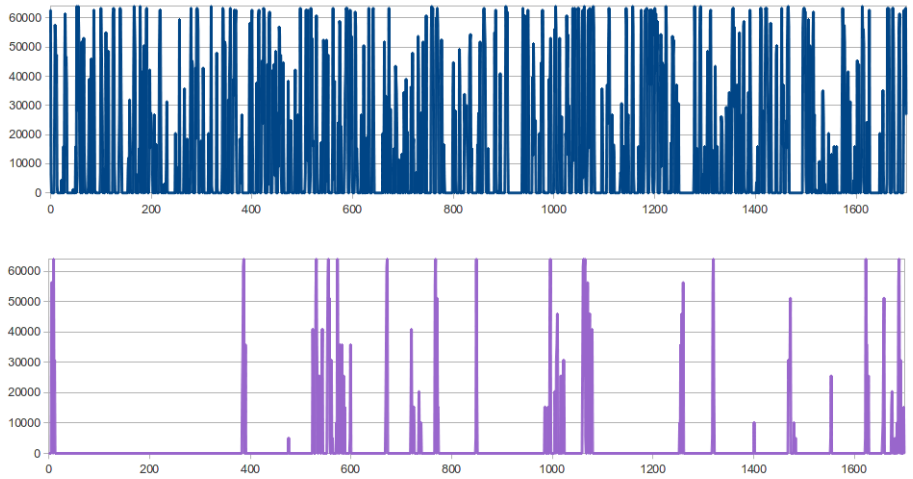

Figura 4.15: Andamento del *Throughput* del dispositivo ricevente del primo e del sesto esperimento, le ascisse sono espresse in secondi mentre le ordinate sono espresse in bit/s.

	80 byte	640 byte
Throughput Minimo (I)	640	5120
Throughput Massimo (I)	64000	64000
Throughput Medio (I)	40176.19210977702	38794.22053231939
Throughput Minimo (R)	0	0
Throughput Massimo (R)	64000	64000
Throughput Medio (R)	21616.7409948542	3791.3307984790877
Rapporto Throughput	0.5380485272419258	0.09772926859867018

Tabella 4.23: Valori medi del *Throughput* del primo e del sesto esperimento espressi in bit/s.

Per quanto riguarda il throughput medio del dispositivo ricevente, quello del sesto esperimento risulta bassissimo rispetto a quello del primo esperimento (circa 21616 bit/s contro circa 3791 bit/s). Questo risultato viene sottolineato ancor più dal rapporto tra i throughput (circa 0,538 del primo esperimento contro circa 0,09 del sesto esperimento). Per quanto riguarda l'andamento dell'intera simulazione mostrato nella figura 4.15, si nota subito come il secondo grafico sia quasi interamente schiacciato a zero e poco denso soprattutto nella prima parte. Possiamo facilmente concludere che il comportamento del secondo grafico è inaccettabile rispetto al primo.

Vediamo ora le statistiche rigardanti gli invii e le ricezioni dei pacchetti nella tabella 4.24 e l'andamento del rapporto d'invio nella figura 4.16. Data la differenza nella dimensione dei pacchetti, sono stati inviati circa un ottavo dei pacchetti nel primo esperimento rispetto al quinto (73506 contro 8593)

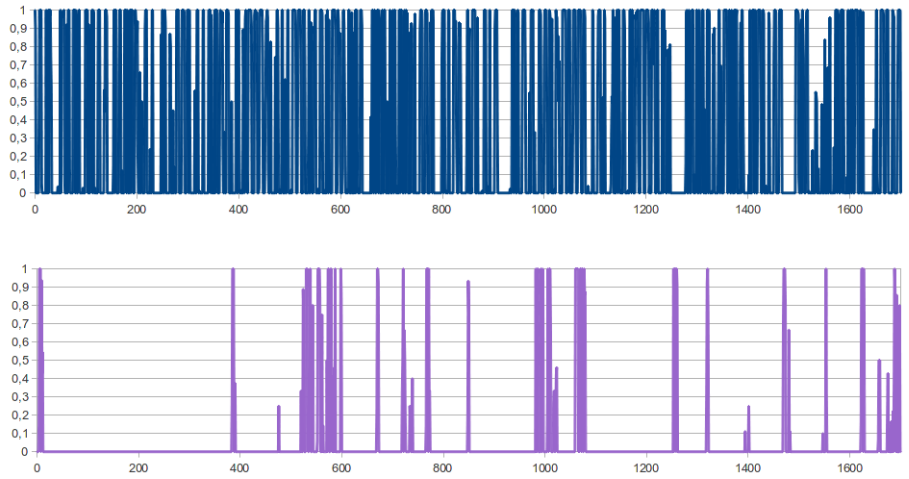

Figura 4.16: Andamento del rapporto d'invio del primo e del sesto esperimento, le ascisse sono espresse in secondi mentre le ordinate vanno da 0 a 1.

	80 byte	640 byte
Pacchetti Inviati	73506	8593
Pacchetti Ricevuti	39513	821
Pacchetti Persi	33993	7772
Rapporto Medio	0.5375479552689576	0.09554288374258117
Numero intervalli vuoti	391	909
Massimo intervallo vuoto	17	19

Tabella 4.24: Statistiche dei pacchetti del primo e del sesto esperimento.

ed ovviamente sono stati ricevuti correttamente molti più pacchetti nel primo esperimento (39513 contro 821) e le perdite sono state maggiori nel primo esperimento (33993 contro 7772). In questo caso dobbiamo affidarci al rapporto di invio per capire meglio i risultati, e questo premia di molto il primo esperimento (circa 0,53 contro circa 0,09). Invece, per quanto riguarda lo studio delle perdite, il numero di intervalli in cui non sono stati ricevuti nessuno dei pacchetti inviati sono maggiori nel sesto esperimento (391 contro 909) e l'intervallo massimo vuoto risulta maggiore sempre nel sesto esperimento seppur di poco (17 contro 19). Per quanto riguarda l'andamento dell'intera simulazione mostrato nella figura 4.16, il secondo grafico risulta poco denso e tendente allo zero a causa delle enormi perdite quindi possiamo concludere che il secondo grafico ha un andamento inaccettabile rispetto al primo.

Riportiamo infine i valori medi della distanza e della velocità nelle tabelle 4.25 e 4.26.

Dopo queste statistiche possiamo concludere che la dimensione dei

	80 byte	640 byte
Distanza Minima	1.7077971554148954	0.6739424862199368
Distanza Massima	105.08872826648025	122.6671130454006
Distanza Media	25.992020109110552	13.53384660832921

Tabella 4.25: Valori medi delle distanze del primo e del sesto esperimento espressi in metri.

	80 byte	640 byte
Velocità Minima (I)	0	0
Velocità Massima (I)	24.93	24.59
Velocità Media (I)	15.05332667913325	14.215920481927718
Velocità Minima (R)	0	0
Velocità Massima (R)	28.5	24.42
Velocità Media (R)	14.965321675847127	13.920722891566308

Tabella 4.26: Valori medi della velocità del primo e del sesto esperimento espressi in metri al secondo.

pacchetti ad 80 byte è largamente preferibile a quella a 640 byte a causa delle enormi perdite di quest'ultima. Possiamo dunque concludere che la configurazione ad 80 byte risulta la migliore in assoluto, la configurazione a 160 byte può essere comunque utilizzata con risultati simili a quella ad 80 byte, mentre le configurazioni a 320 byte e 640 byte danno risultati non accettabili.

4.2.4 Inversione delle posizioni

Studiamo ora il canale di comunicazione cercando di capire se è simmetrico o asimmetrico tramite due confronti entrambi tra due esperimenti svolti con la stessa configurazione (canale di controllo a 3 Mbit/s con pacchetti ad 80 byte) ma con diverse posizioni dei dispositivi: il primo esperimento avviene con le posizioni standard cioè il dispositivo inviante davanti ed il dispositivo ricevente dietro, mentre il secondo esperimento avviene con le posizioni alternate è il dispositivo inviante dietro e quello ricevente davanti. I primi due esperimenti di questo tipo sono stati svolti di seguito con condizione del traffico pressoché identiche mentre gli altri due esperimenti sono stati svolti consecutivamente ma in un altro giorno con condizioni di traffico diverse. Cominciamo con il primo confronto.

Cominciamo confrontando il *Jitter* di cui riportiamo i valori medi nella tabella 4.27 e l'andamento durante tutta la simulazione nella figura 4.17. Il *jitter* minimo del primo esperimento risulta inferiore di un ordine di grandezza rispetto al settimo (ordine E^{-9} contro ordine E^{-8}) mentre quello massimo del settimo esperimento risulta minore rispetto al primo (circa 424 millisecondi contro circa

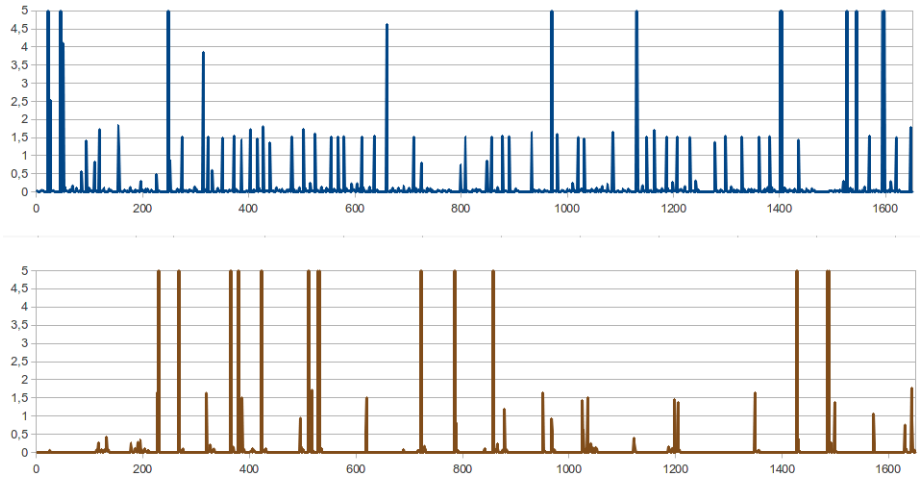

Figura 4.17: Andamento del *jitter* del primo e del settimo esperimento, le ascisse sono espresse in secondi mentre le ordinate sono espresse in millisecondi.

	80 byte	80 byte invertiti
Jitter Minimo	2.00103609250310E-9	5.90472141376438E-8
Jitter Massimo	424.7175441883311	369.54829162561396
Jitter Medio	0.15502994465923736	0.5883503282581366

Tabella 4.27: Valori medi del *Jitter* del primo e del settimo esperimento espressi in millisecondi.

369 millisecondi). Il *jitter* medio premia di poco primo esperimento (circa 0,15 millisecondi contro circa 0,5 millisecondi). Per quanto riguarda l'andamento dell'intera simulazione riportato nella figura 4.17, i due grafici riportati sono molto diversi in quanto il primo ha alcuni picchi e per il resto ha un andamento costante su valori piccoli mentre il secondo ha molti picchi, molti valori nulli a causa delle perdite dei pacchetti e per quanto riguarda i valori medi ha un comportamento simile al primo grafo. Sembra quindi che l'andamento migliore sia quello del primo grafo.

	80 byte	80 byte invertiti
Throughput Minimo (I)	640	640
Throughput Massimo (I)	64000	64000
Throughput Medio (I)	40176.19210977702	39048.5637823372
Throughput Minimo (R)	0	0
Throughput Massimo (R)	64000	64000
Throughput Medio (R)	21616.7409948542	3983.871543264942
Rapporto Throughput	0.5380485272419258	0.1020235101467922

Tabella 4.28: Valori medi del *Throughput* del primo e del settimo esperimento espressi in bit/s.

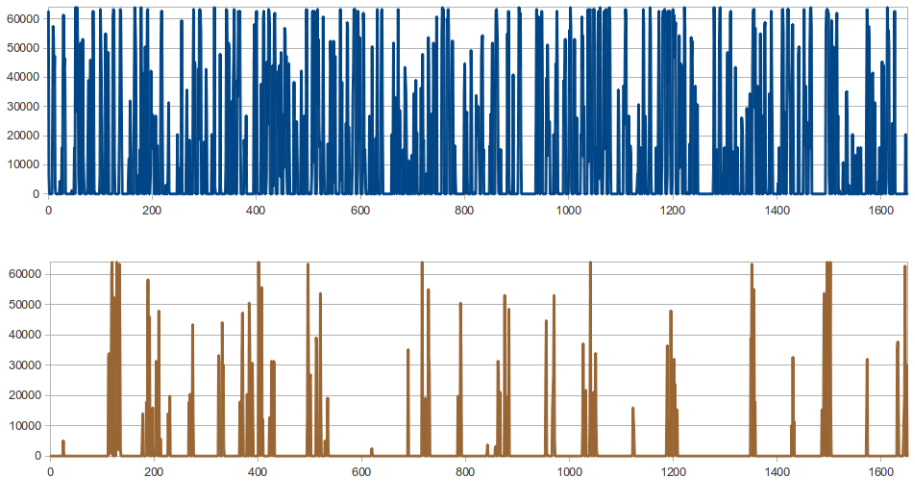

Figura 4.18: Andamento del *Throughput* del dispositivo ricevente del primo e del settimo esperimento, le ascisse sono espresse in secondi mentre le ordinate sono espresse in bit/s.

Proseguiamo il confronto prendendo in considerazione il throughput riportando i valori medi di entrambi i dispositivi nella tabella 4.28 e l'andamento dell'intera simulazione del throughput del dispositivo ricevente nella figura 4.18. I valori minimi e massimo del throughput sono poco interessanti mentre è molto utile analizzare i valori medi. Il throughput medio del dispositivo inviante del primo esperimento è di poco superiore a quello del settimo esperimento (circa 40176 bit/s conto circa 39048 bit/s), facendoci intuire come i periodi di silenzio e parlato totali di questi due esperimenti siano molto simili. Per quanto riguarda il throughput medio del dispositivo ricevente, quello del settimo esperimento risulta molto basso rispetto a quello del primo esperimento (circa 21616 bit/s contro circa 3983 bit/s). Questo risultato viene sottolineato ancor più dal rapporto tra i throughput (circa 0,538 del primo esperimento contro circa 0,10 del settimo esperimento). Per quanto riguarda l'andamento dell'intera simulazione mostrato nella figura 4.18, si nota subito come il secondo grafico abbia molti vuoti sparsi per l'intera durata dell'esperimento e quindi risulta poco denso in confronto al primo. Possiamo facilmente concludere che il comportamento del primo grafico è preferibile al secondo.

Vediamo ora le statistiche rigardanti gli invii e le ricezioni dei pacchetti nella tabella 4.29 e l'andamento del rapporto d'invio nella figura 4.19. Sono stati inviati circa lo stesso numero di pacchetti (73506 contro 74507) e sono stati ricevuti correttamente molti più pacchetti nel primo esperimento (39513 contro 7159) e le perdite sono state di gran lunga maggiori nel settimo esperimento (33993 contro 67349). Coerentemente, il rapporto premia il primo esperimento

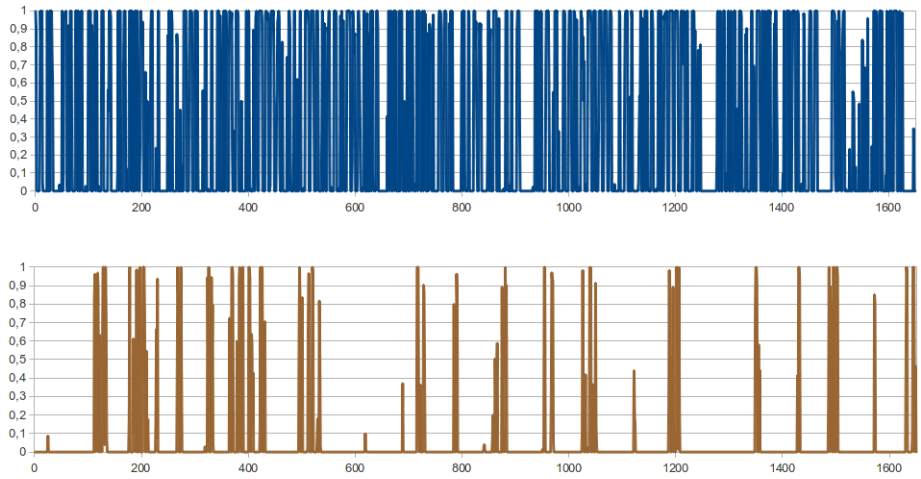

Figura 4.19: Andamento del rapporto d'invio del primo e del settimo esperimento, le ascisse sono espresse in secondi mentre le ordinate vanno da 0 a 1.

	80 byte	80 byte invertiti
Pacchetti Inviati	73506	74507
Pacchetti Ricevuti	39513	7159
Pacchetti Persi	33993	67349
Rapporto Medio	0.5375479552689576	0.09608493161716349
Numero intervalli vuoti	391	924
Massimo intervallo vuoto	17	19

Tabella 4.29: Statistiche dei pacchetti del primo e del settimo esperimento.

(circa 0,53 contro circa 0,09). Infine, per quanto riguarda lo studio delle perdite, il numero di intervalli in cui non sono stati ricevuti nessuno dei pacchetti inviati sono maggiori nel settimo esperimento (391 contro 924) e l'intervallo massimo vuoto risulta maggiore sempre nel settimo esperimento seppur di poco (17 contro 19). Per quanto riguarda l'andamento dell'intera simulazione mostrato nella figura 4.19, il secondo grafico risulta poco denso e tendente allo zero a causa delle enormi perdite quindi possiamo concludere che il secondo grafico ha un andamento inaccettabile rispetto al primo.

	80 byte	80 byte invertiti
Distanza Minima	1.7077971554148954	0.30294125413138917
Distanza Massima	105.08872826648025	69.33354087999494
Distanza Media	25.992020109110552	12.134883528359937

Tabella 4.30: Valori medi delle distanze del primo e del settimo esperimento espressi in metri.

	80 byte	80 byte invertiti
Velocità Minima (I)	0	0
Velocità Massima (I)	24.93	20.14
Velocità Media (I)	15.05332667913325	12.574111312849144
Velocità Minima (R)	0	0
Velocità Massima (R)	28.5	22.0
Velocità Media (R)	14.965321675847127	12.620898044692618

Tabella 4.31: Valori medi della velocità del primo e del settimo esperimento espressi in metri al secondo.

Riportiamo infine i valori medi della distanza e della velocità nelle tabelle 4.30 e 4.31.

Dopo queste statistiche possiamo concludere che la posizione standard e quella invertita hanno dato risultati molto diversi e la prima è largamente preferibile alla seconda. Questo risultato implica l'asimmetria del canale, vediamo un altro confronto al riguardo per vedere se la tesi viene confermata o meno.

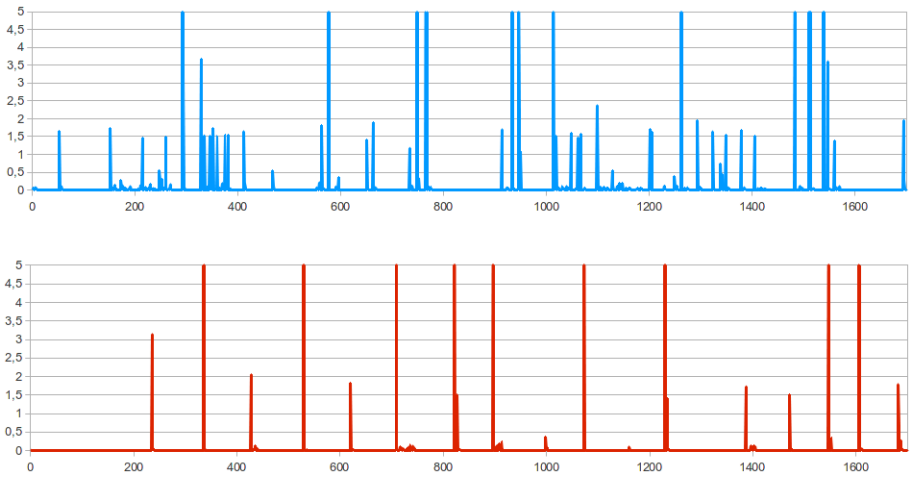

Figura 4.20: Andamento del *jitter* dell'ottavo e del nono esperimento, le ascisse sono espresse in secondi mentre le ordinate sono espresse in millisecondi.

	80 byte	80 byte invertiti
Jitter Minimo	6.357230882505E-8	8.69356476737492E-6
Jitter Massimo	571.9321211923734	399.57503738375743
Jitter Medio	0.38132807363576593	0.8950034538337555

Tabella 4.32: Valori medi del *Jitter* dell'ottavo e del nono esperimento espressi in millisecondi.

Cominciamo confrontando il *Jitter* di cui riportiamo i valori medi nella tabella 4.32 e l'andamento durante tutta la simulazione nella figura 4.20. Il *jitter* minimo dell'ottavo esperimento risulta inferiore di due ordini di grandezza rispetto al settimo (ordine E^{-8} contro ordine E^{-6}) mentre quello massimo del nono esperimento risulta minore rispetto all'ottavo (circa 571 millisecondi contro circa 399 millisecondi). Il *jitter* medio premia di poco l'ottavo esperimento (circa 0,38 millisecondi contro circa 0,89 millisecondi). Per quanto riguarda l'andamento dell'intera simulazione riportato nella figura 4.20, i due grafici riportati sembrano uguali per quanto riguarda i picchi però il primo ha un andamento migliore per quanto riguarda i valori medi in quanto il secondo si annulla molte volte a causa delle perdite. Sembra quindi che l'andamento migliore sia quello del primo grafo.

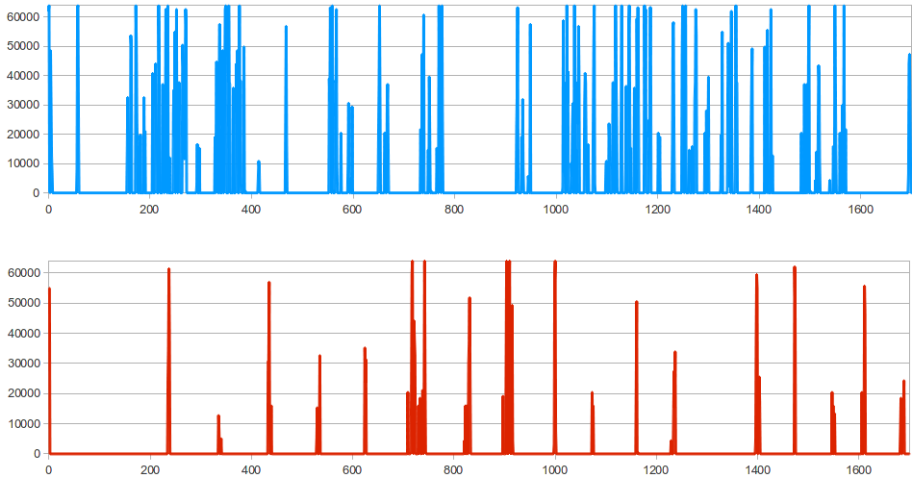

Figura 4.21: Andamento del *Throughput* del dispositivo ricevente dell'ottavo e del nono esperimento, le ascisse sono espresse in secondi mentre le ordinate sono espresse in bit/s.

	80 byte	80 byte invertiti
Throughput Minimo (I)	640	640
Throughput Massimo (I)	64000	64000
Throughput Medio (I)	38336.83426443203	38420.281690140844
Throughput Minimo (R)	0	0
Throughput Massimo (R)	64000	64000
Throughput Medio (R)	8650.130353817505	2357.1830985915494
Rapporto Throughput	0.22563496751329004	0.06135257199835768

Tabella 4.33: Valori medi del *Throughput* dell'ottavo e del nono esperimento espressi in bit/s.

Proseguiamo il confronto prendendo in considerazione il throughput ripor-

tando i valori medi di entrambi i dispositivi nella tabella 4.33 e l'andamento dell'intera simulazione del throughput del dispositivo ricevente nella figura 4.21. I valori minimi e massimo del throughput sono poco interessanti mentre è molto utile analizzare i valori medi. Il throughput medio del dispositivo inviante è pressoché identico nei due esperimenti (circa 38336 bit/s conto circa 38420 bit/s), facendoci intuire come i periodi di silenzio e parlato totali di questi due esperimenti siano uguali. Per quanto riguarda il throughput medio del dispositivo ricevente, quello del nono esperimento risulta molto basso rispetto a quello dell'ottavo esperimento (circa 8650 bit/s contro circa 2357 bit/s) ed entrambi sono risultano molto bassi. Questo risultato viene sottolineato ancor più dal rapporto tra i throughput (circa 0,22 dell'ottavo esperimento contro circa 0,06 del nono esperimento). Per quanto riguarda l'andamento dell'intera simulazione mostrato nella figura 4.21, si nota subito come entrami i grafi siano poco densi ed affetti da vuoti importanti. Il secondo grafico ha molti vuoti sparsi per l'intera durata dell'esperimento. Possiamo facilmente concludere che il comportamento del primo grafico è preferibile al secondo.

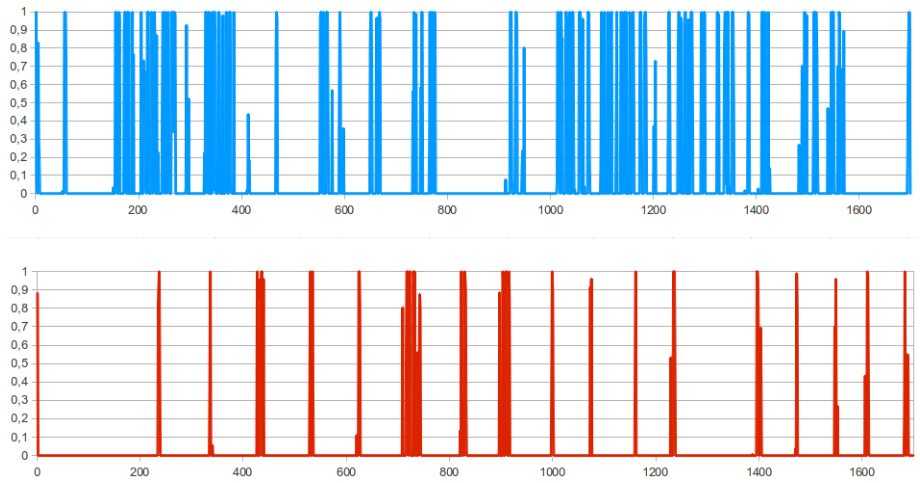

Figura 4.22: Andamento del rapporto d'invio dell'ottavo e del nono esperimento, le ascisse sono espresse in secondi mentre le ordinate vanno da 0 a 1.

	80 byte	80 byte invertiti
Pacchetti Inviati	69529	73678
Pacchetti Ricevuti	15152	4285
Pacchetti Persi	54377	69393
Rapporto Medio	0.21792345639948799	0.058158473357040094
Numero intervalli vuoti	756	1034
Massimo intervallo vuoto	25	23

Tabella 4.34: Statistiche dei pacchetti dell'ottavo e del nono esperimento.

Vediamo ora le statistiche rigardanti gli invii e le ricezioni dei pacchetti nella tabella 4.34 e l'andamento del rapporto d'invio nella figura 4.22. Sono stati inviati un maggior numero di pacchetti nel nono esperimento (69529 contro 73678) e sono stati ricevuti correttamente molti più pacchetti nell'ottavo esperimento (15152 contro 4285) e le perdite sono state di gran lunga maggiori nel nono esperimento (54377 contro 69393) in entrambi i casi le perdite sono molto alte. Coerentemente il rapporto premia l'ottavo esperimento (circa 0,21 contro circa 0,05). Infine, per quanto riguarda lo studio delle perdite, il numero di intervalli in cui non sono stati ricevuti nessuno dei pacchetti inviati sono maggiori nel nono esperimento (756 contro 1034) e l'intervallo massimo vuoto risulta maggiore nell'ottavo esperimento seppur di poco (25 contro 23). Per quanto riguarda l'andamento dell'intera simulazione mostrato nella figura 4.22, entrambi i grafici risultano poco densi e tendenti allo zero a causa delle enormi perdite quindi possiamo concludere che entrambi i grafici hanno un andamento inaccettabile anche se il primo risulta migliore.

	80 byte	80 byte invertiti
Distanza Minima	0.33375210733557875	0.637635616612288
Distanza Massima	99.51684796426862	51.07209465782002
Distanza Media	12.269350289037057	6.966671532125965

Tabella 4.35: Valori medi delle distanze dell'ottavo e del nono esperimento espressi in metri.

	80 byte	80 byte invertiti
Velocità Minima (I)	0	0
Velocità Massima (I)	24.34	24.14
Velocità Media (I)	14.541226788432411	8.346844682835778
Velocità Minima (R)	0	0
Velocità Massima (R)	24.179	23.4
Velocità Media (R)	14.514052809211824	7.713029617537167

Tabella 4.36: Valori medi della velocità dell'ottavo e del nono esperimento espressi in metri al secondo.

Riportiamo infine i valori medi della distanza e della velocità nelle tabelle 4.35 e 4.36.

Dopo queste statistiche possiamo concludere che anche questa volta la posizione standard e quella invertita hanno dato risultati molto diversi e che la prima è largamente preferibile alla seconda. Questo risultato conferma l'asimmetria del canale.

4.2.5 Il Beaconing

Vediamo ora se la presenza del beaconing, cioè la presenza di altri messaggi inviati nel canale, influisce sugli esperimenti o meno. Cominciamo confrontando due esperimenti entrambi effettuati sul canale di controllo a 3 Mbit/s con pacchetti di 80 byte, il primo senza il beaconing mentre il secondo con il beaconing attivo:

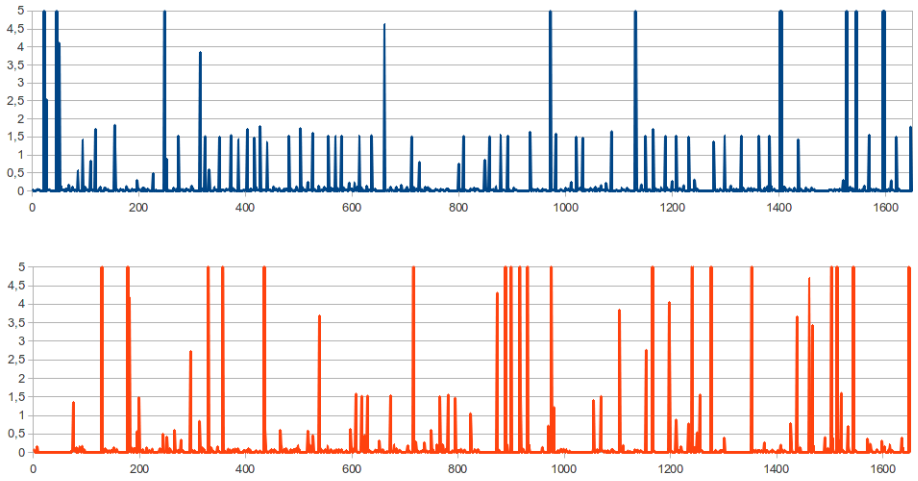

Figura 4.23: Andamento del *Jitter* del primo e del decimo esperimento, le ascisse sono espresse in secondi mentre le ordinate sono espresse in millisecondi.

	80 byte	80 byte beaconing
Jitter Minimo	2.00103609250310E-9	6.895338514900E-8
Jitter Massimo	424.7175441883311	353.83943712871417
Jitter Medio	0.15502994465923736	0.2599204832173714

Tabella 4.37: Valori medi del *Jitter* del primo e del decimo esperimento espressi in millisecondi.

Il primo parametro preso in considerazione è il *Jitter* di cui riportiamo i valori medi nella tabella 4.37 e l'andamento durante tutta la simulazione nella figura 4.23. Notiamo subito che il *jitter* minimo del primo esperimento è più piccolo di quello del decimo (E^{-9} contro E^{-8}) mentre il picco massimo premia il decimo esperimento (circa 424 millisecondi contro 353), quello che importa però è il *jitter* medio che premia, seppur di poco, il primo esperimento (circa 0,15 millisecondi contro circa 0,25 millisecondi). Per quanto riguarda l'andamento dell'intera simulazione riportato nella figura 4.23, i due grafici riportati sono poco simili, e si può notare come il secondo grafico abbia un maggior numero di picchi e sia più denso soprattutto nei valori molto piccoli (sotto *0,5*).

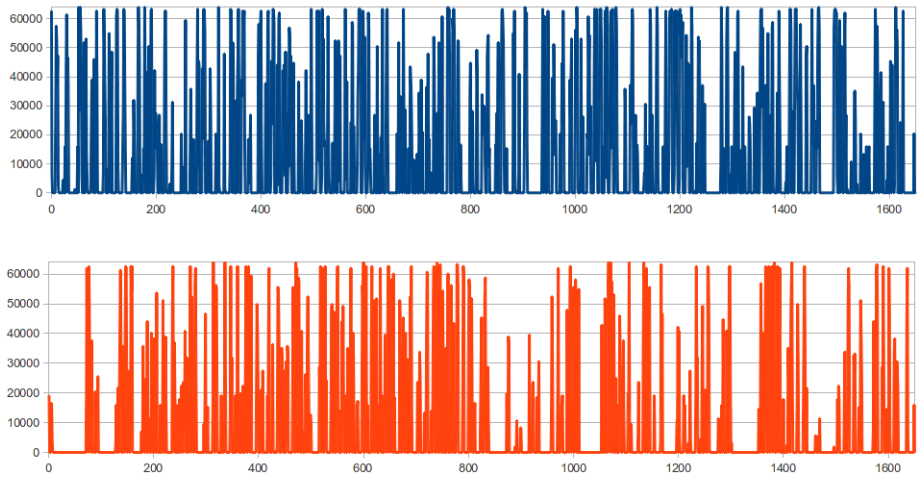

Figura 4.24: Andamento del *Throughput* del dispositivo ricevente del primo e del decimo esperimento, le ascisse sono espresse in secondi mentre le ordinate sono espresse in bit/s.

	80 byte	80 byte beaconing
Throughput Minimo (I)	640	640
Throughput Massimo (I)	64000	64000
Throughput Medio (I)	40176.19210977702	39729.140767824494
Throughput Minimo (R)	0	0
Throughput Massimo (R)	64000	64000
Throughput Medio (R)	21616.7409948542	15205.557586837294
Rapporto Throughput	0.5380485272419258	0.3827305925315114

Tabella 4.38: Valori medi del *Throughput* del primo e del decimo esperimento espressi in bit/s.

Proseguiamo prendendo in considerazione il throughput riportando i valori medi di entrambi i dispositivi nella tabella 4.38 e l'andamento dell'intera simulazione del throughput del dispositivo ricevente nella figura 4.24. Vediamo subito come i valori minimi e massimo del throughput siano poco interessanti mentre è molto utile analizzare i valori medi. Il throughput medio dei due dispositivi è molto vicino (circa 40176 bit/s conto circa 39729 bit/s). Nonostante i valori simili del throughput di invio medio del dispositivo inviante, il throughput medio del dispositivo ricevente premia il primo esperimento (circa 21616 bit/s contro circa 15205 bit/s). Questo risultato viene sottolineato ancor più dal rapporto tra i throughput che tiene in considerazione il throughput d'invio e quello di ricezione dei dispositivi (circa 0,53 del primo esperimento contro circa 0,38 del decimo esperimento). Per quanto riguarda l'andamento dell'intera simulazione mostrato nella figura 4.24, si può subito notare come il primo grafico sia più

denso rispetto al secondo, soprattutto nella fase iniziale ed in alcuni punti sparsi verso la fine, e di conseguenza possiamo concludere che nel primo esperimento il *throughput* si comporta meglio rispetto al decimo.

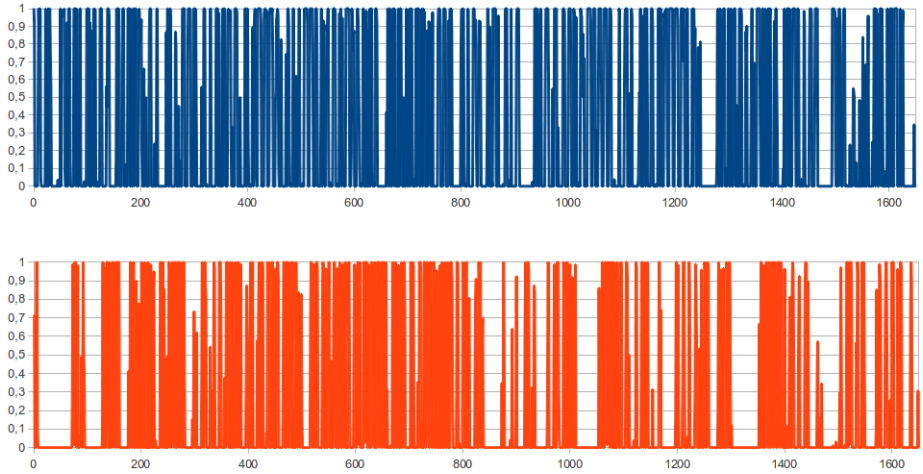

Figura 4.25: Andamento del rapporto d'invio del primo e del decimo esperimento, le ascisse sono espresse in secondi mentre le ordinate vanno da 0 a 1.

	80 byte	80 byte beaconing
Pacchetti Inviati	73506	68302
Pacchetti Ricevuti	39513	26074
Pacchetti Persi	33993	42229
Rapporto Medio	0.5375479552689576	0.3817457761119733
Numero intervalli vuoti	391	494
Massimo intervallo vuoto	17	13

Tabella 4.39: Statistiche dei pacchetti del primo e del decimo esperimento.

Vediamo ora le statistiche riguardanti gli invii e le ricezioni dei pacchetti nella tabella 4.39 e l'andamento del rapporto d'invio nella figura 4.25. Sono stati inviati molti più pacchetti nel primo esperimento (73506 contro 68302) e sono stati ricevuti correttamente più pacchetti nel primo esperimento (39513 contro 26074) e, di conseguenza, le perdite sono state maggiori nel decimo esperimento (33993 contro 42229). Infatti il rapporto di invio risulta maggiore nel primo esperimento (circa 0,53 contro circa 0,38). Coerentemente con queste statistiche il numero di intervalli in cui non sono stati ricevuti nessuno dei pacchetti inviati sono maggiori nel decimo esperimento (391 contro 494) mentre l'intervallo massimo vuoto risulta maggiore nel primo esperimento (17 contro 13). Per quanto riguarda l'andamento dell'intera simulazione mostrato nella figura 4.25, essendo il primo grafico più denso rispetto al secondo, soprattutto nella parte iniziale,

possiamo concludere che l'andamento medio del primo grafico risulta migliore di quello del secondo.

	80 byte	80 byte beaconing
Distanza Minima	1.7077971554148954	0.12438214704267671
Distanza Massima	105.08872826648025	40.567919394316654
Distanza Media	25.992020109110552	14.66422243374262

Tabella 4.40: Valori medi delle distanze del primo e del decimo esperimento espressi in metri.

	80 byte	80 byte beaconing
Velocità Minima (I)	0	0
Velocità Massima (I)	24.93	25.14
Velocità Media (I)	15.05332667913325	15.941189647921968
Velocità Minima (R)	0	0
Velocità Massima (R)	28.5	26.67
Velocità Media (R)	14.965321675847127	16.03920853243602

Tabella 4.41: Valori medi della velocità del primo e del decimo esperimento espressi in metri al secondo.

Riportiamo infine i valori medi della distanza e della velocità nelle tabelle 4.40 e 4.41.

Dopo queste statistiche possiamo concludere che in questo primo caso il beaconing sembra aver influito poco sull'esperimento. Anche se l'esperimento senza beaconing ha dato migliori risultati, questi possono essere indipendenti dal beaconing, infatti due esperimenti consecutivi con le stesse impostazioni possono dare risultati anche molto diversi.

Vediamo ora un altro confronto simile a quello appena mostrato ma con la dimensione dei pacchetti a 160 byte.

	160 byte	160 byte beaconing
Jitter Minimo	6.7242939282579E-8	8.422297680044E-8
Jitter Massimo	437.1474631242738	435.1438948276788
Jitter Medio	0.29301558752997603	0.6411757379499315

Tabella 4.42: Valori medi del *Jitter* del quarto e dell'undicesimo esperimento espressi in millisecondi

Il primo parametro preso in considerazione è il *Jitter* di cui riportiamo i valori medi nella tabella 4.42 e l'andamento durante tutta la simulazione nella figura 4.26. Notiamo subito che il *jitter* minimo e quello massimo sono quasi identici

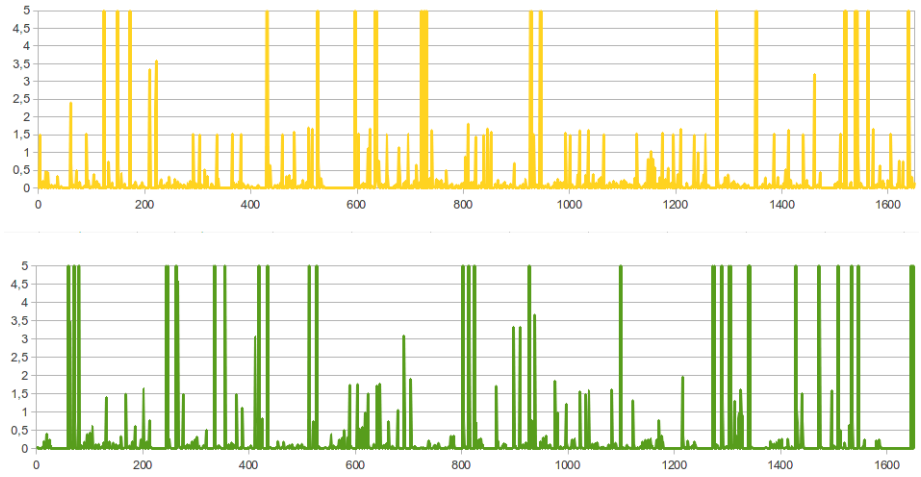

Figura 4.26: Andamento del *Jitter* del quarto e dell'undicesimo esperimento, le ascisse sono espresse in secondi mentre le ordinate sono espresse in millisecondi.

(ordine E^{-8} per il minimo, circa 437 millisecondi contro 435 per il massimo) invece il *jitter* medio premia, il quarto esperimento (circa 0,29 millisecondi contro circa 0,64 millisecondi). Per quanto riguarda l'andamento dell'intera simulazione riportato nella figura 4.26, i due grafici riportati hanno un andamento molto simile sia per quanto riguarda i valori piccoli sia per quanto riguarda i picchi, anche se sono distribuiti in maniera differente.

Figura 4.27: Andamento del *Throughput* del dispositivo ricevente del quarto e dell'undicesimo esperimento, le ascisse sono espresse in secondi mentre le ordinate sono espresse in bit/s.

Proseguiamo prendendo in considerazione il throughput riportando i valori medi di entrambi i dispositivi nella tabella 4.43 e l'andamento dell'intera simula-

	160 byte	160 byte beaconing
Throughput Minimo (I)	1280	1280
Throughput Massimo (I)	64000	64000
Throughput Medio (I)	39381.70434782609	41623.132530120485
Throughput Minimo (R)	0	0
Throughput Massimo (R)	64000	64000
Throughput Medio (R)	21054.33043478261	18965.115848007415
Rapporto Throughput	0.5346221242439658	0.4556388405962322

Tabella 4.43: Valori medi del *Throughput* del quarto e dell'undicesimo esperimento espressi in bit/s.

zione del throughput del dispositivo ricevente nella figura 4.27. Vediamo subito come i valori minimi e massimo del throughput siano poco interessanti mentre è molto utile analizzare i valori medi. Il throughput medio dei due dispositivi è molto vicino (circa 39381 bit/s conto circa 41623 bit/s). Nonostante i valori simili del throughput di invio medio del dispositivo inviante, il throughput medio del dispositivo ricevente premia di poco il quarto esperimento (circa 21054 bit/s contro circa 18965 bit/s). Questo risultato viene sottolineato ancor più dal rapporto tra i throughput che tiene in considerazione il throughput d'invio e quello di ricezione dei dispositivi (circa 0,53 del quarto esperimento contro circa 0,45 dell'undicesimo esperimento). Per quanto riguarda l'andamento dell'intera simulazione mostrato nella figura 4.27, si può subito notare come il primo grafico sia più denso rispetto al secondo, soprattutto nella fase finale e di conseguenza possiamo concludere che nel quarto esperimento il *throughput* si comporta meglio rispetto all'undicesimo.

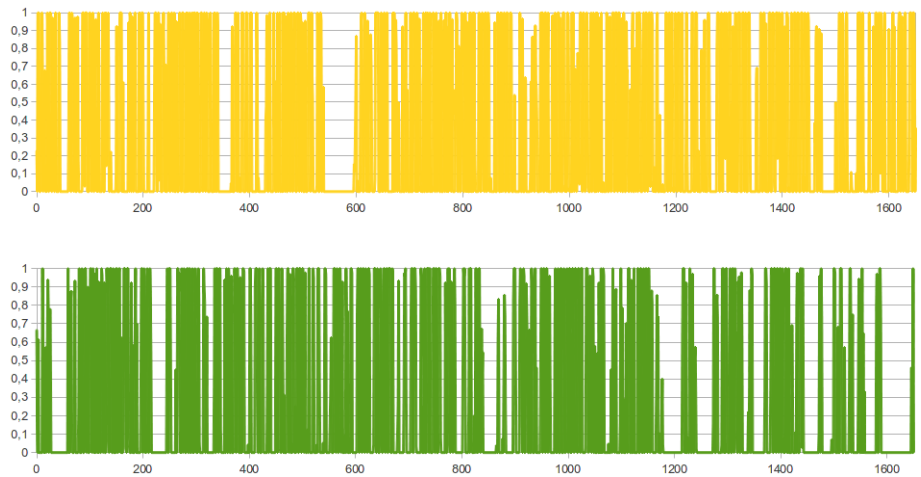

Figura 4.28: Andamento del rapporto d'invio del quarto e dell'undicesimo esperimento, le ascisse sono espresse in secondi mentre le ordinate vanno da 0 a 1.

	160 byte	160 byte beaconing
Pacchetti Inviati	38556	35325
Pacchetti Ricevuti	19967	16046
Pacchetti Persi	18589	19279
Rapporto Medio	0.5178701110073659	0.4542392073602265
Numero intervalli vuoti	329	417
Massimo intervallo vuoto	15	10

Tabella 4.44: Statistiche dei pacchetti del quarto e dell'undicesimo esperimento.

Vediamo ora le statistiche riguardanti gli invii e le ricezioni dei pacchetti nella tabella 4.44 e l'andamento del rapporto d'invio nella figura 4.28. Sono stati inviati più pacchetti nel quarto esperimento (38556 contro 35325) e sono stati ricevuti correttamente più pacchetti sempre nel quarto esperimento (19967 contro 16046), di conseguenza, le perdite sono state maggiori nell'undicesimo esperimento (18589 contro 19279). Infatti il rapporto di invio risulta maggiore nel quarto esperimento (circa 0,51 contro circa 0,45). Coerentemente con queste statistiche il numero di intervalli in cui non sono stati ricevuti nessuno dei pacchetti inviati sono maggiori nell'undicesimo esperimento (329 contro 417) mentre l'intervallo massimo vuoto risulta maggiore nel quarto esperimento (15 contro 10). Per quanto riguarda l'andamento dell'intera simulazione mostrato nella figura 4.28, essendo il primo grafico più denso rispetto al secondo, soprattutto nella parte finale, possiamo concludere che l'andamento medio del primo grafico risulta migliore di quello del secondo.

	160 byte	160 byte beaconing
Distanza Minima	0.4848244257621118	0.12438214704267671
Distanza Massima	92.95214940084924	40.567919394316654
Distanza Media	23.421475432946607	14.66422243374262

Tabella 4.45: Valori medi delle distanze del quarto e dell'undicesimo esperimento espressi in metri.

	160 byte	160 byte beaconing
Velocità Minima (I)	0	0
Velocità Massima (I)	24.97	25.67
Velocità Media (I)	15.32631058600146	16.349123855781507
Velocità Minima (R)	0	0
Velocità Massima (R)	26.56	28.62
Velocità Media (R)	15.261059099765257	16.421023600473173

Tabella 4.46: Valori medi della velocità del quarto e dell'undicesimo esperimento espressi in metri al secondo.

Riportiamo infine i valori medi della distanza e della velocità nelle tabelle

4.45 e 4.46.

Dopo queste statistiche possiamo concludere che anche in questo caso il beaconing sembra aver influito poco sull'esperimento, ancor meno che nel caso precedente. Di conseguenza possiamo concludere che il beaconing non influisce sensibilmente nella qualità degli esperimenti.

4.2.6 Frequenza a 16 Kbit/s

Concludiamo cercando di capire se abbassando la frequenza di invio da circa 64 Kbit/s a circa 16 Kbit/s otteniamo dei miglioramenti o meno.

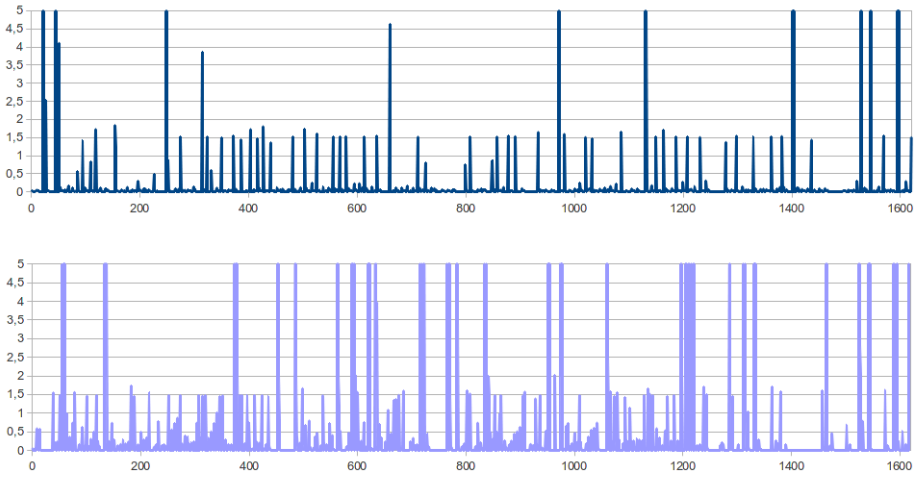

Figura 4.29: Andamento del *Jitter* del primo e del dodicesimo esperimento, le ascisse sono espresse in secondi mentre le ordinate sono espresse in millisecondi.

	64 Kbit/s	16 Kbit/s
Jitter Minimo	2.0010360925031E-9	4.50139146530125E-6
Jitter Massimo	424.7175441883311	862.1318009206494
Jitter Medio	0.15502994465923736	1.5620411160058738

Tabella 4.47: Valori medi del *Jitter* del primo e del dodicesimo esperimento espressi in millisecondi.

Il primo parametro preso in considerazione è il *Jitter* di cui riportiamo i valori medi nella tabella 4.47 e l'andamento durante tutta la simulazione nella figura 4.29. Notiamo subito che il *jitter* minimo minore è quello del primo esperimento (ordine E^{-9} contro ordine E^{-6}), il *jitter* massimo del dodicesimo esperimento risulta doppio rispetto a quello del primo esperimento (circa 424 contro circa 862) infine il *jitter* medio premia di gran lunga il primo esperimento

(circa 0,15 millisecondi contro circa 1.56 millisecondi). Per quanto riguarda l'andamento dell'intera simulazione riportato nella figura 4.29, possiamo vedere come il secondo grafico sia costantemente più denso del primo e che quindi sia preferibile al secondo.

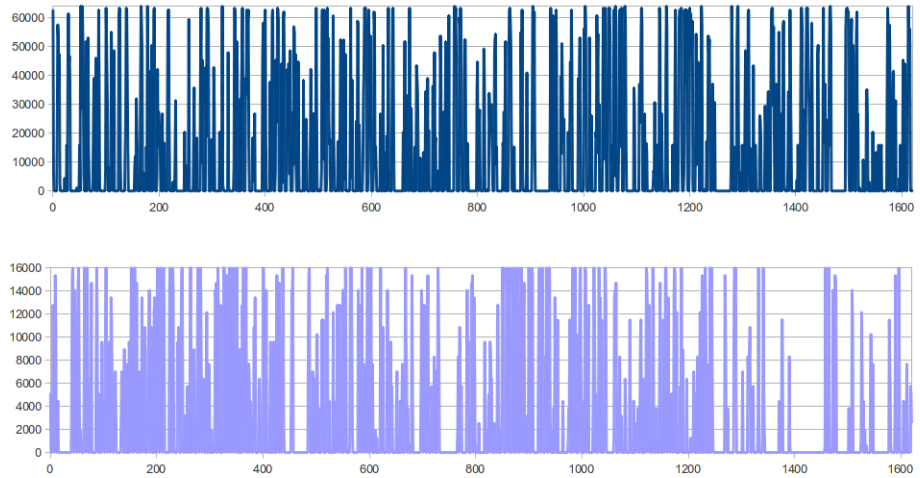

Figura 4.30: Andamento del *Throughput* del dispositivo ricevente del primo e del dodicesimo esperimento, le ascisse sono espresse in secondi mentre le ordinate sono espresse in bit/s.

	64 kbit/s	16 Kbit/s
Throughput Minimo (I)	640	640
Throughput Massimo (I)	64000	16000
Throughput Medio (I)	40176.19210977702	10358.821218074656
Throughput Minimo (R)	0	0
Throughput Massimo (R)	64000	64000
Throughput Medio (R)	21616.7409948542	5113.713163064833
Rapporto Throughput	0.5380485272419258	0.4936578260605693

Tabella 4.48: Valori medi del *Throughput* del primo e del dodicesimo esperimento espressi in bit/s.

Proseguiamo prendendo in considerazione il throughput riportando i valori medi di entrambi i dispositivi nella tabella 4.48 e l'andamento dell'intera simulazione del throughput del dispositivo ricevente nella figura 4.30. Vediamo subito come il confronto di questi valori ha poco senso dal momento che il throughput è basato sulla frequenza d'invio che in questo caso è molto diversa. Ha però senso confrontare il rapporto tra i throughput che premia di poco il primo esperimento (circa 0,53 circa 0,49). Per quanto riguarda l'andamento dell'intera simulazione mostrato nella figura 4.30, si può subito notare come il primo grafico sia più

denso rispetto al secondo nella parte finale mentre il secondo grafico risulta più denso rispetto al primo nella parte iniziale.

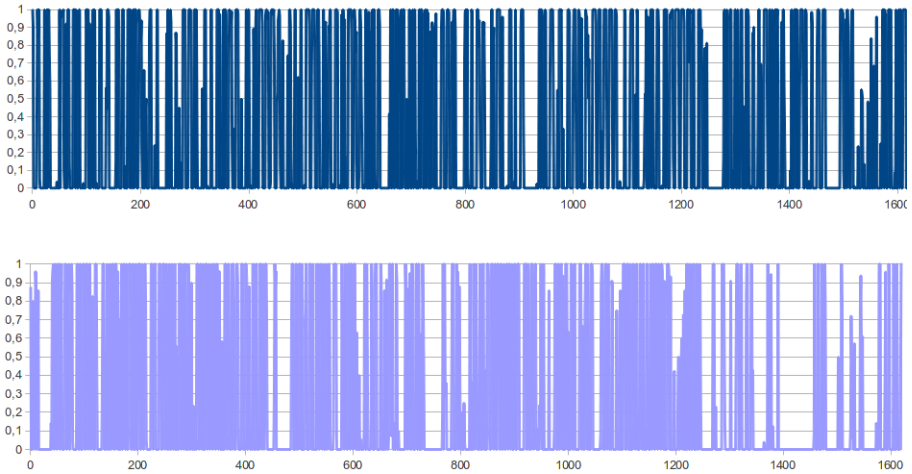

Figura 4.31: Andamento del rapporto d'invio del primo e del dodicesimo esperimento, le ascisse sono espresse in secondi mentre le ordinate vanno da 0 a 1.

	64 Kbit/s	16 Kbit/s
Pacchetti Inviati	73506	16587
Pacchetti Ricevuti	39513	8160
Pacchetti Persi	33993	8427
Rapporto Medio	0.5375479552689576	0.4919515283052993
Numero intervalli vuoti	391	342
Massimo intervallo vuoto	17	14

Tabella 4.49: Statistiche dei pacchetti del primo e del dodicesimo esperimento.

Vediamo ora le statistiche riguardanti gli invii e le ricezioni dei pacchetti nella tabella 4.49 e l'andamento del rapporto d'invio nella figura 4.31. Anche in questo caso gli unici dati confrontabili sono il rapporto di invio che premia il primo esperimento (circa 0,53 contro circa 0,49) e le statistiche relative agli intervalli vuoti che invece vanno a favore del dodicesimo esperimento (391 contro 342 per il numero di intervalli vuoti e 17 contro 14 per il massimo intervallo). Per quanto riguarda l'andamento dell'intera simulazione mostrato nella figura 4.31, il secondo grafo risulta mediamente più denso rispetto al primo, ma soffre anche di alcuni vuoti soprattutto nella parte finale.

Riportiamo infine i valori medi della distanza e della velocità nelle tabelle 4.50 e 4.51.

	64 Kbit/s	16 Kbit/s
Distanza Minima	1.7077971554148954	0.3834526207558863
Distanza Massima	105.08872826648025	115.63899875015989
Distanza Media	25.992020109110552	14.778734757230735

Tabella 4.50: Valori medi delle distanze del primo e del dodicesimo esperimento espressi in metri.

	64 Kbit/s	16 Kbit/s
Velocità Minima (I)	0	0
Velocità Massima (I)	24.93	25.39
Velocità Media (I)	15.05332667913325	15.932057995839932
Velocità Minima (R)	0	0
Velocità Massima (R)	28.5	26.71
Velocità Media (R)	14.965321675847127	15.917988131653066

Tabella 4.51: Valori medi della velocità del primo e del dodicesimo esperimento espressi in metri al secondo.

Dopo queste statistiche possiamo concludere che l'aver abbassato la frequenza di invio non ha portato benefici rispetto al caso a 64 Kbit/s.

4.2.7 Riassunto dei Risultati

Vediamo in questa sezione un riassunto di tutti i risultati più importanti di tutti e dodici gli esperimenti svolti. I risultati vengono mostrati nella tabella 4.52.

Num.	Jitter Medio	Throughput (R)	Rapporto d'invio
1	0.1550299446	21616.740994	0.5375479552
2	0.2027442371	15293.812445	0.3616339289
3	0.2328372734	14122.628262	0.3483988736
4	0.2930155875	21054.330434	0.5178701110
5	5.2841140529	2177.3913043	0.0526798231
6	13.893848009	3791.3307984	0.0955428837
7	0.5883503282	3983.8715432	0.0960849316
8	0.3813280736	8650.1303538	0.2179234563
9	0.8950034538	2357.1830985	0.0581584733
10	0.259920483	15205.557586	0.3817457761
11	0.641175737	18965.115848	0.4542392073
12	1.562041116	5113.7131630	0.4919515283

Tabella 4.52: I risultati più importanti degli esperimenti svolti.

4.2.8 Il Throughput Reale

Un parametro che abbiamo preso in considerazione, ma che non abbiamo riportato in precedenza, è il throughput reale. Con throughput reale intendiamo il valore del throughput del dispositivo ricevente al netto dei periodi di vuoto, cioè al netto di quegli intervalli dove non sono stati ricevuti nessuno dei pacchetti inviati. Questo parametro ci aiuta a capire il comportamento del throughput del dispositivo ricevente quando il sistema di invio funziona correttamente. La tabella 4.53 contiene i valori del throughput reale del dispositivo ricevente di tutti gli esperimenti.

Num.	Throughput Reale Ricevente
1	32503.74064516129 bit/s
2	30594.243859649123 bit/s
3	32340.451546391752 bit/s
4	29464.943970767355 bit/s
5	24450.602040816328 bit/s
6	27748.244755244756 bit/s
7	22659.89340101523 bit/s
8	29196.474842767297 bit/s
9	26233.71568627451 bit/s
10	27718.4 bit/s
11	30899.817220543806 bit/s
12	7684.73224852071 bit/s

Tabella 4.53: I valori del throughput reale del dispositivo ricevente.

4.2.9 L'influenza della Distanza sui risultati

Vediamo ora quanto la distanza abbia influito sui parametri più importanti di tutti gli esperimenti. Per farlo calcoliamo il *correlation index* che è un valore che va da -1 a 1 dove più i valori sono vicini agli estremi e più c'è correlazione tra i parametri mentre più i valori sono vicini allo zero e più la correlazione è inesistente. I risultati sono riportati nella tabella 4.54.

Come è possibile vedere dalla tabella, la distanza influisce poco sul correlation index di jitter, throughput e rapporto d'invio. I valori degli esperimenti, escludendone alcuni, sono molto vicini a zero e i valori medi soprattutto per quanto riguarda throughput e rapporto d'invio sono molto bassi.

4.2.10 L'influenza della Velocità sui risultati

Concludiamo analizzando quanto la velocità del dispositivo ricevente abbia influito sui parametri più importanti di tutti gli esperimenti con la stessa tecnica adottata nel paragrafo precedente. I risultati sono riportati nella tabella 4.55.

Num.	Dist/Jitt	Dist/Thr	Dist/Rapp
1	-0,03	-0,003	-0,002
2	-0,07	0,07	0,04
3	-0,06	-0,12	-0,11
4	-0,01	0,01	-0,03
5	0,05	0,01	-0,01
6	0,53	-0,20	0,01
7	0,41	-0,19	-0,17
8	0,65	-0,06	-0,05
9	0,58	-0,20	-0,02
10	-0,05	0,02	0,03
11	0,06	0,004	-0,02
12	0,40	-0,04	-0,10
Media	**0,20**	**-0,05**	**-0,03**

Tabella 4.54: I valori del correlation index relativi alla distanza.

Num.	Vel/Jitt	Vel/Thr	Vel/Rapp
1	0,02	-0,01	-0,006
2	-0,02	0,03	0,02
3	0,001	0,001	-0,07
4	0,04	-0,05	-0,07
5	-0,01	0,07	0,001
6	0,17	-0,001	-0,01
7	0,07	-0,09	-0,03
8	0,12	0,16	0,06
9	0,20	-0,11	-0,03
10	0,01	0,06	0,04
11	0,02	0,04	0,07
12	-0,01	0,008	0,002
Media	**0,05**	**0,009**	**-0,001**

Tabella 4.55: I valori del correlation index relativi alla velocità.

Possiamo fare lo stesso ragionamento del paragrafo precedente. La velocità del dispositivo ricevente non influisce sul correlation index del jitter, throughput e rapporto d'invio infatti i valori dei vari esperimenti e i valori medi risultano molto bassi.

Capitolo 5

Conclusioni

L'obiettivo primario di questa tesi era quello di rispondere alla domanda se fosse possibile realizzare un'applicazione *VoIP* per sistemi veicolari che sfrutti come hardware le *LinkBird-MX* comunicanti tramite interfacce wireless a corto raggio. Per rispondere a questa domanda sono stati realizzati nel linguaggio di programmazzione *Java* un programma per simulare una tale applicazione *VoIP* ed un programma di Post-Processing per analizzare i risultati ottenuti, il tutto incorporato in una semplice ed intuitiva interfaccia grafica. Accanto a questo obiettivo primario ce ne erano altri secondari come lo studio del canale di comunicazione e lo studio delle interferenze tramite il beaconing. Grazie alle molte misurazioni effettuate e grazie all'analisi dei risultati ottenuti possiamo rispondere a queste domande.
Innanzitutto non è possibile realizzare un'applicazione *VoIP* con l'hardware messo a disposizione principalmente a causa delle troppe perdite. Se le perdite non fossero così alte gli altri parametri presi in considerazione come il *jitter* ed il *throughput* sarebbero accettabili. Possiamo inoltre concludere che il canale risulta asimmetrico e questa risulta essere un'altra limitazione per lo sviluppo di un'applicazione di questo tipo. Infine possiamo concludere che il beaconing, introdotto per simulare la coabitazione con un'altra applicazione che competa per il canale, in realtà non influisce sui risultati e, allo stesso modo, che l'uso di una codifica diversa che impieghi una frequenza di invio inferiore non influisce sui risultati.

5.1 Estensioni

L'esito negativo dei risultati ottenuti a seguito degli esperimenti svolti per capire se fosse possibile la realizzazione di un'applicazione *VoIP* nell'ambito dei sistemi veicolari con l'hardware fornito può spronare ad ulteriori studi al ri-

guardo. Un primo passo per un'eventuale estensione potrebbe essere quello del cambio dell'hardware. Anziché sfruttare le *LinkBird-MX* si potrebbero cercare altri dispositivi diversi che svolgano lo stesso compito ma con i quali le perdite siano minori. A tal proposito anche il solo cambio dell'antenna con una più alta potrebbe influire positivamente sui risultati. Tutto il software realizzato può essere utilizzato anche con altri hardware senza alcun tipo di modifiche in modo da poter testare il nuovo hardware facilmente. Un'ulteriore modifica che potrebbe essere realizzata è quella di estendere il *C2X-SDK* fornito insieme alle *LinkBird-MX* in modo che i pacchetti inviati sfruttino la pila protocollare *C2X* anziché sfruttare il classico livello *IP*.

5.2 Esperienze Acquisite

Questa tesi mi ha permesso di conoscere i sistemi veicolari, capire il loro funzionamento e i loro problemi. Ho inoltre migliorato ed ampliato le mie conoscenze riguardo le reti wireless il *Voice over IP* e lo studio di parametri come *Jitter* e *Throughput* fondamentali per queste tematiche. Tutto il software da me sviluppato è stato realizzato tramite il linguaggio di programmazione *Java*, quindi le mie abilità e conoscenze al riguardo sono aumentate.

Ringraziamenti

Possono bastare poche righe per ringraziare tutte le persone che in tutti questi anni mi hanno aiutato e supportato in questa esperienza? La risposta è **ASSOLUTAMENTE NO**.
Ci vorrebbero pagine per elencare tutte le persone che ho incontrato durante questo cammino e spiegare perché la loro presenza mi è stata di così grande aiuto ed in cosa ognuno di loro è stato originale e speciale per me. Infatti anche le persone che non ho potuto frequentare in questo periodo, o più in generale in quest'ultimo anno, negli anni passati mi hanno aiutato tantissimo e hanno reso i momenti difficili molto più facili da superare ed hanno accresciuto i momenti di felicità.
Dal momento quindi che non posso e non voglio scrivere paginate di ringraziamenti, ringrazierò ora le persone che mi sono sempre state accanto e quelle che in quest'ultimo periodo mi sono state più vicino sperando che chi non verrà menzionato sappia che ha avuto e sempre avrà un posto nei miei ricordi.
Comincio ringraziando tutta la mia famiglia che mi è sempre stata vicina e mi ha sempre lasciato libero di prendere ogni decisione che volessi ed in particolar modo ringrazio **mamma Daniela**, **nonna Liliana**, **zia Luciana**, **zio Antonio** e **Lorenzo**. Ringrazio gli amici con cui ho passato momenti a dir poco fantastici ed in particolar modo **la stellina Carmela**, **la vecchia Irene**, **l'omone Leo**, **il prof LoRe**, **il mitico Becciu** e **le sorelle Manfrè**.
Ringrazio i ricercatori del *C.N.R.* che mi hanno aiutato e supportato per tutta la durata della tesi: **il dottor Paolo Santi** ideatore di questa tesi, **la dottoressa Maria Elena Renda** che ha avuto la pazienza di autarmi durante gli esperimenti in laboratorio e soprattutto **la dottoressa Francesca Martelli** che ha avuto l'infinita pazienza di guidare insieme a me in ogni esperimento su strada.
Concludo infine ringraziando il mio amore, **Nadia**, una delle persone più speciali che io abbia mai incontrato la cui sola presenza mi ha dato forze e stimoli incredibili per migliorarmi, è anche grazie a lei se ho raggiunto questo traguardo.

Bibliografia

[1] Network Division NEC Laboratories: *CAR-2-X Communication SDK. User Guide*, NEC Europe Ltd., 29.10.2010 Version 1.5.2.

[2] A. Festag, R. Baldessari, L. Long, W. Zhang, A. Sarma and A. Fukukawa: *CAR-2-X Communication for Safety and Infotainment in Europe*, NEC Technical Journal, Vol. 3, No. 1, Special Issue: ITS, March 2008.

[3] Car-2-Car Communication Consortium. *C2C-CC Manifesto*. Versione 1.1, Agosto 2007.

[4] Car-2-Car Communication Consortium official website *http://www.car-to-car.org/*.

[5] D. Jiang and L. Delgrossi: *IEEE 802.11p: Towards an International Standard for Wireless Access in Vehicular Environments*, in Proc. of IEEE Vehicular Technology Conference (VTC) Spring 2008.

[6] Stefan Mangold, Sunghyun Choi, Peter May, Ole Klein, Guido Hiertz, Lothar Stibor: *IEEE 802.11e Wireless LAN for Quality of Service*, Philips Research Germany, Germany.

[7] Ian Dangerfield, David Malone, Douglas J. Leith: *Understanding 802.11e Voice Behaviour via Testbed Measurements and Modeling*, Proc. Workshop on Wireless Network Measurement (WiNMee 2007, LNCS 4427).

[8] Ian Dangerfield: *Wireless Network Measurement - VoIP and 802.11e*, Master thesis 2007.

[9] Andreas Festag, Roberto Baldessari, Wenhui Zhang and Long Le: *CAR-2-X Communication SDK - A Software Toolkit for Rapid Application Development and Experimentations*, IEEE VehiMobil 2009.

[10] RFC - 2330: *Framework for IP Performance Metrics.*

[11] RFC - 2679: *A One-way Delay Metric for IPPM.*

[12] RFC - 3393: *IP Packet Delay Variation Metric for IP Performance Metrics (IPPM).*

[13] RFC - 3550: *RTP: A Transport Protocol for Real-Time Applications.*

[14] A. Estepa, R. Estepa, and J. Vozmediano: *A New Approach for VoIP Traffic Characterization*, IEEE Communication Letters. v8 i10. 644-646.

[15] Athina P. Markopoulou, Member, IEEE, Fouad A. Tobagi, Fellow, IEEE, Mansour J. Karam, Member, IEEE: *Assessing the Quality of Voice Communications over Internet Backbones*, IEEE Transactions on Networking, Vol. 11 No. 5, October 2003.

[16] Sriram, K. Whitt, W.: *Characterizing superposition arrival processes in packet multiplexers for voice and data*, IEEE Journal on Selected Areas in Communications, vol. SAC-4, No. 6, September 1986, pp. 833-846.

[17] R. Gurin, H. Ahmadi, and M. Naghshineh: *Equivalent capacity and its application to bandwidth allocation in high-speed networks*, IEEE J. Select. Area Commun., vol. 9, pp. 968-981, Sept. 1991.

[18] Gyung-Ho Hwang, Student Member IEEE, and Dong-Ho Cho, Senior Member IEEE: *New Access Scheme for VoIP Packets in IEEE 802.11e Wireless LANs*, IEEE Communications Letters 9(7), 667669 (2005).

[19] Bur Goode, Senior Member IEEE: *Voice Over Internet Protocol (VoIP)*, Proceedings of the IEEE, Vol. 9.

[20] Francesca Martelli, M. Elena Renda, and Paolo Santi: *A Measurement-based Study of Beaconing Performance in IEEE 802.11p Vehicular Networks*, IIT - CNR, Pisa Italy.

[21] Wi-Fi Alliance: *http://www.wi-fi.org/index.php.*

[22] 802.11 Standards: *http://standards.ieee.org/about/get/802/802.11.html.*

[23] 802.11p: *http://standards.ieee.org/getieee802/download/802.11p-2010.pdf.*

[24] F. Martelli, M.E. Renda, P.Santi: *Measuring IEEE 802.11p Performance for Active Safety Applications in Cooperative Vehicular Systems*, IEEE Vehicular Technology Conference (VTC - Spring), to appear.

[25] Skype: *http://www.skype.com/intl/it/welcomeback/.*

[26] Martin Müller: *WLAN 802.11p Measurements for Vehicle to Vehicle (V2V) DSRC Application Note.* In: Rohde & Schwarz (2009).

[27] Alexander Paier, Johan Karedal, Nicolai Czink, Charlotte Dumard, Thomas Zemen, Fredrik Tufvesson, Andreas F. Molisch, Christoph F. Mecklenbräuker: *Comparison of Lund07 vehicular channel measurements with the IEEE 802.11p channel model*, temporary document (08) 436 presented at COST 2100 Meeting, Wroclaw, Poland, Feb. 6-8, 2008.

[28] V. Shivaldova, G. Maier, D. Smely, N. Czink, A. Alonso, A. Winkelbauer, A. Paier, C.F. Mecklenbräuker: *Performance Evaluation of IEEE 802.11p Infrastructure-to-Vehicle Tunnel Measurements*, in Proc. The 7th International wireless Communications and Mobile Computing Conference (IWCMC-2011), pages 1 - 5, Istanbul, Turkey, July, 2011.

[29] Christoph Sommer, David Eckhoff, Reinhard German and Falko Dressler: *A Computationally Inexpensive Empirical Model of IEEE 802.11p Radio Shadowing in Urban Environments*, in 8th IEEE/IFIP Conference on Wireless On demand Network Systems and Services (WONS 2011), Poster Session. Bardonecchia, Italy: IEEE, January 2011, pp. 8490.

[30] Sangho Shin, Henning Schulzrinne: *Experimental Measurement of the Capacity for VoIP Traffic in IEEE 802.11 WLANs*, in Proc. IEEE ICC07, June 2007.

[31] Anna Sfairopoulou, Carlos Macián, Boris Bellalta: *Dynamic measurement-based codec selection for VoIP in multirate IEEE 802.11 WLANs*, Technical Report TD(07)018, Cost290 (http://www.cost290.org), February 2007.

[32] Sangho Shin, Henning Schulzrinne: *Towards the Quality of Service for VoIP traffic in IEEE 802.11 Wireless Networks*, Computer Science Technical Report Series 2008.