Università degli Studi di Pisa

FACOLTÀ DI SCIENZE MATEMATICHE, FISICHE E NATURALI
Dipartimento di Informatica

Tesi di Laurea Magistrale in Informatica

Studio sperimentale delle prestazioni di un'applicazione di beaconing per reti veicolari multihop

Candidato:
Alessandro Franchini

Relatori:
Dott. Paolo Santi
Prof. Maurizio A. Bonuccelli

Controrelatore:
Prof. Susanna Pelagatti

Anno Accademico 2011–2012

a mamma e papà

Indice

Elenco delle figure

Elenco delle tabelle

Capitolo 1

Introduzione

Il processo globale di motorizzazione e urbanizzazione degli ultimi vent'anni ha portato ad un costante aumento del traffico stradale, con la conseguente necessità di sviluppare applicazioni orientate a migliorare e a rendere sicura la guida. L'idea è di riuscire a far comunicare tra di loro i veicoli senza la necessità di alcun punto di accesso fisso: ogni auto è un'entità indipendente in grado di trasmettere e ricevere informazioni che, opportunamente processate, permettono di garantire una maggiore sicurezza agli utenti della strada. Questo tipo di struttura comunicativa prende il nome *Vehicular Ad-Hoc Network o VANET* [1].

Le *VANET* nascono come forma particolare delle *Mobile Ad-Hoc Network o MANET* [2], in cui dispositivi mobili comunicano tra di loro attraverso un canale wireless. La comunicazione avviene tra nodi paritetici in maniera cooperativa andando a formare configurazioni dinamiche e sempre in mutamento, da cui il termine *Ad-hoc*. A differenza delle *MANET*, le reti veicolari sono influenzate da diversi fattori, tra cui velocità e densità dei nodi, che possono variare notevolmente in funzione del tipo di strada (autostrada, extra-urbana, urbana, etc.). Inoltre lo spazio utilizzato dai nodi ha una conformazione molto particolare, cioè le strade, e non tutto lo spazio circolare o rettangolare come usualmente si suppone nelle MANET. La presenza di questi fattori, da un lato comporta la necessità di riuscire a far comunicare molti veicoli in tempi ragionevoli, senza incorrere in problemi di congestione; dall'altro garantisce la capillarità della rete ed una propagazione a lunga distanza delle informazioni.

Le principali tipologie di applicazioni realizzabili spaziano dalla *sicurezza attiva*, ai *servizi di pubblica assistenza*, all'*ausilio alla guida*. Alcuni esempi possono essere la segnalazione di condizioni stradali pericolose e l'indicazione di avvenuto incidente. Nei due casi, dal momento in cui il veicolo identifi-

ca la situazione di rischio, intraprende il doppio processo di segnalazione al conducente e propagazione ai nodi vicini. Nel secondo scenario è possibile immaginare ad esempio un avvertimento sul cruscotto dell'auto, l'accensione delle luci di emergenza e l'invio di un messaggio che, una volta ricevuto dalle altre auto, accende automaticamente le luci di emergenza.

Per disporre di maggiori finanziamenti per i costi di sviluppo e incoraggiare una rapida adozione di questi sistemi, è stato reso possibile lo sviluppo di *servizi privati*, i quali vanno a ricoprire applicazioni nell'ambito del *business* e dell'*entertainment*. Esempi sono soluzioni per l'impresa che permettono la gestione delle flotte oppure la possibilità di effettuare chiamate gratuite tra veicoli tramite il servizio *VoiP (Voice over IP)* [3].

Prima di poter implementare le applicazioni sopra descritte bisogna prendere in considerazione tutti i problemi che scaturiscono dall'utilizzo della tecnologia wireless in un ambiente così particolare. La ricerca sta focalizzando le attenzioni su problematiche relative al congestionamento della rete [4], alla perdita di pacchetti [5] e ad altre misure in funzione della distanza e della velocità dei veicoli, alcune delle quali studio di questa tesi.

Un ultimo aspetto da non sottovalutare, riguarda i possibili abusi ai quali le VANET si prestano: violazione della privacy mediante intercettazione, sabotaggio alla viabilità mediante falsi avvisi di emergenza e simulazione di mezzi di emergenza per sorpassare altri veicoli sono solo alcune di una serie di azioni mirate a manomettere l'integrità della comunicazione. La soluzione proposta a questo problema, prevede la firma e la certificazione dei messaggi prodotti attraverso un *Hardware Security Module*.

1.1 Descrizione della Tesi

Questa tesi si presenta come proseguimento del lavoro descritto nell'articolo di ricerca *A Measurement-based Study of Beaconing Performance in IEEE 802.11p Vehicular Networks* [6], realizzato dai ricercatori del *Consiglio Nazionale delle Ricerche (C.N.R.) di Pisa* sotto la guida del *Dott. Paolo Santi*. Tale studio, presentato alla *IEEE INFOCOM 2012*, ha previsto inizialmente la scrittura di un applicativo per lo scambio di *beacon* tra **due veicoli** (paradigma *singlehop*); successivamente, è stato testato su circa 1000 km per raccogliere dati su cui tracciare diversi tipi di analisi descritte in seguito.

Il lavoro svolto in questa tesi, ha avuto come scopo primario quello di estendere tale applicativo, considerando uno scenario con **tre (o più) veicoli**

per valutare l'impatto delle performance nel caso di comunicazioni non più *singlehop*, ma *multihop*. Dopo un iniziale conoscenza sul precedente studio e sugli strumenti hardware (dispositivo *LinkBird-MX*) e software (librerie Java *C2X*) da utilizzare, il lavoro si è articolato in due fasi.

La prima fase, ha previsto la modifica dell'applicazione permettendo ad ogni veicolo di memorizzare le informazioni dei suoi vicini e di inserirle in ogni pacchetto inviato. Tutti gli eventi generati da ogni invio e ricezione, vengono registrati in file di log il quale conterrà i dati utili nella fase di analisi. La seconda fase, è stata caratterizzata dalla scelta delle misure da analizzare e la conseguente stesura del programma di post-processing. I risultati generati da quest'ultimo, hanno permesso di realizzare grafici e tabelle, alcune delle quali di seguito discusse e analizzate. Il tutto è stato testato inizialmente in laboratorio e poi per strada, sperimentando situazioni di traffico reale. Da sottolineare l'esclusività dei test sul campo, dato che quasi tutti gli studi di ricerca effettuati nel panorama mondiale, sono frutto di valutazioni empiriche.

1.2 Obiettivi della Tesi

Il principale obiettivo della tesi è stato quello di calcolare le performance della propagazione delle informazioni tra più veicoli in modalità multihop, facendo riferimento alla performance ottenuta in modalità singlehop. In particolare sono stati misurati il *beacon (packet) delivery rate (PDR)* globale e in funzione della distanza, il *packet inter-reception (PIR) time* e il PIR blackout. Tutte queste misure sono state considerate sia nel link *singlehop* che *multihop*.

Un altro obiettivo è stato valutare le misure sopra citate secondo il modello L/N (Gilbert-Elliot model [7], [8]), il quale mostra l'effetto delle condizioni di *LineOfSight (LOS) / NonLineOfSight (NLOS)* sulla qualità del canale. A differenza degli esperimenti precedenti, nei quali i due veicoli difficilmente si trovavano in una condizione di NonLineOfSight, grazie all'impiego di tre auto è stato possibile caratterizzare ed evidenziare le differenze delle due modalità.

Infine l'ultimo importante obiettivo è stato quello di verificare quanto potesse influire nelle comunicazioni, la differenza di altezza tra i veicoli. L'attenzione è stata posta su possibili asimmetrie nel canale e sulla consistenza dei benefici dell'auto alta come mezzo propagatore posto tra due auto basse.

I risultati ottenuti hanno pienamente soddisfatto le aspettative e hanno permesso di raggiungere gli obiettivi prefissati. Inoltre, hanno messo in evi-

denza ulteriori interrogativi da prendere in considerazione per eventuali studi futuri.

1.3 Stato dell'arte

Per quanto riguardo lo stato dell'arte delle tecnologie utilizzate che stanno alla base di questa tesi, possiamo dire che le reti veicolari sono un campo in cui i test sulla tecnologia e sulle problematiche annesse sono in costante evoluzione, con l'ottica di riuscire nell'arco di pochi anni a garantirne una rapida diffusione.

Facendo una rapida panoramica su studi nell'ambito delle VANET troviamo [9], nel quale sono state calcolate le performance considerando le probabilità di collisione, il throughput e il ritardo. In [10] e [4] invece sono state studiate le problematiche relative al possibile congestionamento della rete wireless.

Per quanto riguarda invece misurazioni su campo, possiamo citare i seguenti lavori. Nell'articolo [11], viene spiegato come effettuare misurazioni su ambienti veicolari con le apparecchiature di prova *Rode & Schwarz (R&S)*. Invece nell'articolo [12] vengono valutate le performance a seguito di misurazioni in uno scenario *vehicle-to-infrastructure (V2I)* all'interno di un tunnel nell'autostrada S1 vicino a Vienna. Nell'articolo [13] viene presentata un simulazione realistica in un ambiente urbano, stimando gli effetti delle costruzioni e di altri ostacoli sulla comunicazione wireless. Nell'articolo [14] vengono presentati i risultati di misurazioni per valutare le performance della comunicazione wireless in differenti scenari: *vehicle-to-vehicle V2V* e *vehicle-to-infrastructure (V2I)*. L'autore ha concentrato le valutazioni nella comunicazione multi-hop, identificando nella distanza e nel *LineOfSight* i due principali fattori che influenzano le prestazioni del canale comunicativo. Infine nell'articolo [15], l'autore misura la qualità del canale durante la guida in ambienti autostradali, urbani e suburbani. Lo studio ha riportato che l'area sub-urbana è la più consona per la comunicazione tra veicoli.

Molteplici invece sono i lavori legati agli argomenti di questa tesi. Per quanto riguarda le applicazioni di sicurezza basate su beaconing possiamo citare l'articolo [16] nel quale sono state effettuate misurazioni sulle reti veicolari concentrandosi sullo scambio di messaggi di beaconing tra veicoli contenenti informazioni di stato per le applicazioni di sicurezza. In [17], sono stati studiati i requisiti di comunicazione per applicazioni basate su conoscenza cooperativa, in particolare prendendo in considerazione tre modelli di applicazioni per

la sicurezza attiva. Nell'articolo [18] sono state studiate, per una certa classe di applicazioni di sicurezza attiva chiamate *Cooperative Collision Warning (CCW)*, le latenze e le probabilità di successo su situazioni reali. Inoltre sono stati esplorati due problemi rilevanti per questo tipo di applicazioni, cioè le performance in funzione della distanza e possibili miglioramenti per il broadcast. Sulla stessa classe di applicazioni, in [19] è stato definito un protocollo con politiche di controllo della congestione, meccanismi per la differenziazione dei servizi e metodi per la rapida propagazione di segnali di emergenza. Le simulazioni hanno dimostrato il raggiungimento di basse latenze nella consegna dei messaggi ed un efficiente uso del canale in scenari stradali complessi. Per quanto riguarda studi su performance di misure oggetto di questa di tesi, troviamo [20] dove l'autore considera un applicazione di *Intersection Collision Warning (ICW)* e valuta PDR e RSSI in funzione della distanza dei due veicolo dall'incrocio e [21] dove invece l'autore presenta un'analisi del PDR in differenti scenari sulla base sia di fattori ambientali incontrollabili che di parametri del canale radio invece controllabili. Infine in [22] è stato analizzata la portata e la frequenza dei Beacon in funzione della velocità e della densità dei veicoli.

1.4 Organizzazione della Tesi

La struttura della tesi è la seguente. Il Capitolo 2 riassume e descrive le principali tecnologie utilizzate in questa tesi. Il Capitolo 3 mostra le peculiarità degli esperimenti effettuati, in particolare il setup fisico utilizzato e i tipi di report redatti in fase di analisi. Il Capitolo 4 descrive accuratamente i vari software sviluppati per lo studio delle performance. I Capitoli 5 e 6 descrivono tutti i risultati ottenuti, prima in maniera generale e poi nel dettaglio. Infine nel capitolo 7 vengono trattate le conclusioni della tesi soffermandosi sulle esperienze acquisite e sui possibili lavori futuri.

Capitolo 2

Le tecnologie utilizzate

Questo capitolo descrive in maniera generale le principali caratteristiche delle reti veicolari e le componenti primarie che ne formano la struttura. Poi passeremo alla presentazione dello sviluppo della tecnologia wireless *IEEE 802.11p*, nata sulla base dello standard *IEEE 802.11*, appositamente realizzata per ambienti veicolari; di questa ne analizzeremo anche, molto brevemente, la pila protocollare. Infine verrà approfondito il concetto di *beaconing* e i tipi di applicazioni che utlizzano tale modalità di comunicazione.

2.1 Le reti veicolari

Le *Vehicular Ad-Hoc Network o VANET* sono una tecnologia che, attraverso l'utilizzo delle automobili, permette di creare una rete mobile. Il loro aspetto fondamentale, che le contraddistingue dalle reti mobili classiche, è l'assenza di alcuna infrastruttura predisposta: infatti è difficile pensare che tutte le strade possano essere coperte da punti di accesso fissi per permettere la comunicazione tra i veicoli. Un tale approccio architetturale, simile a quello dei sistemi cellulari, presenterebbe sia problemi di fattibilità che di sostenibilità economica. Per questi motivi è più sensato immaginare che i veicoli possano da soli costituire l'infrastruttura di comunicazione.

Adottando però questa soluzione, si presentano delle problematiche relative all'instradamento dei pacchetti: ciascun veicolo, attraverso un dispositivo wireless, ha la capacità di comunicare a distanza limitata, generalmente in un range compreso tra i 100 e i 300 metri. Inoltre la topologia della rete subisce rapidi cambiamenti a seguito dei movimenti a velocità elevate dei veicoli, e di conseguenza lo stato della connettività fra i nodi è in continua evoluzione. Diventa così non banale trovare un modo efficiente di far giungere i dati a

destinazione. Tale problema, non è stato oggetto di interesse della tesi, ma bensì a partire da un protocollo di comunicazione esistente, ne è stata valutata l'efficienza in una condizione di comunicazione multihop.

2.1.1 La struttura delle VANET

La struttura di una VANET, come mostrato nella figura 2.1, è composta da tre distinti domini [23]:

- In-vehicle
- Ad-Hoc
- Infra-structure

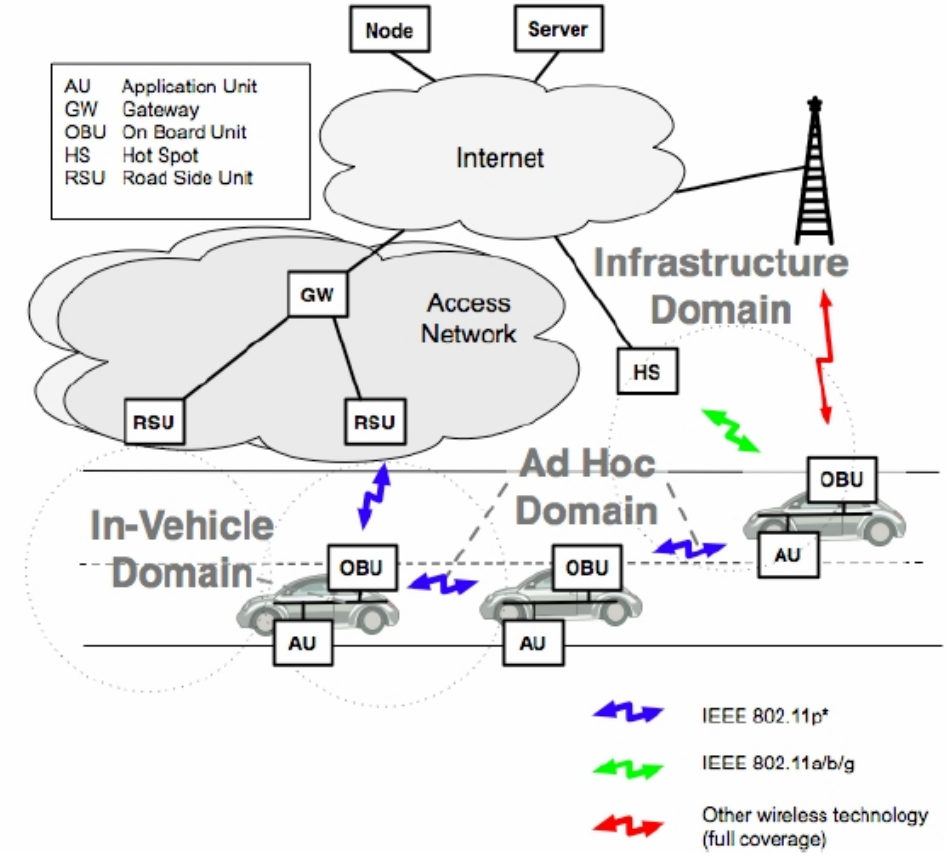

Figura 2.1: Architettura di una VANET

Il dominio nel veicolo (**In-vehicle domain**) è costituito da una *On Board Unit (OBU)* e da una o più *Application Units (AUs)*. Una AU è un dispositivo che esegue una o più applicazioni sfruttando la OBU per la comunicazione: può essere già integrata col veicolo, quindi essere costantemente connessa alla

OBU, oppure può essere portabile come ad esempio un laptop o un tablet. Le due unità solitamente sono connesse tramite rete cablata, ma la connessione può essere anche wireless, ad esempio tramite Bluetooth. E' importante sottolineare che la divisione in due componenti è solo logica: possono infatti risiede in unico componente fisico.

Il dominio ad-hoc (**Ad-Hoc domain**) costituisce il cuore delle rete veicolari, cioè l'insieme dei componenti che si occupano della trasmissione dei dati. Possiamo identificare due unità preposte a questo scopo:le OBUs e le *Road Side Units (RSUs)*. Le OBUs, dotate di antenna wireless, sono i nodi principali della rete: seguono gli standard protocolli di comunicazione che vanno dal broadcast all'unicast, formando così link sia singlehop che multihop. Per migliorare l'efficienza delle VANET in funzione della sicurezza stradale, le OBUs sono state integrate con le RSUs: quest'ultime sono anch'esse dei dispotivi wireless ma statici. La loro presenza non stravolge la struttura base della rete, in quanto non sono dei nodi di coordinamento. Hanno il compito di eseguire particolari applicazioni utili a monitorare lo stato del traffico stradale, ma anche di aumentare la coperturà della rete ricevendo e inoltrando i pacchetti alle OBU.

Il dominio delle infrastrutture (**Infra-structure domain**) è la parte di rete non indispensabile ai fini della comunicazione e della sicurezza tra i veicoli ma ha lo scopo di integrare la rete internet tradizionale. L'integrazione avviene attraverso degli *hotspots (HS)*, pubblici o privati, oppure tramite l'appoggio alle risorse della rete cellulare (GSM, GPRS, UMTS, HSDPA, WiMax, 4G) purché la OBU dotata di un'antenna alternativa per la comunicazione wireless.

Questa tesi non spazia in tutti e tre i domini, ma va ad interessare direttamente il primo e parte del secondo. Infatti, come vedremo in seguito, l'analisi ha riguardato uno scenario in cui ciascun veicolo è formato ad una OBU collegata da una AU, eseguita su un portatile. Le tre OBU poi si scambiano le informazioni, costituendo link sia singlehop che multihop, senza però l'ausilio di alcuna RSUs.

2.1.2 L'architettura a livelli delle VANET

L'architettura comunicativa delle componenti di una VANET, OBUs e RSUs, viene mostrata nella figura 2.2. Analizziamo brevemente le caratteristiche dei vari livelli:

- L'*Application Layer* fornisce le principali funzionalità necessarie all'esecuzione di un processo applicativo, in particolare l'invio, la ricezione

e il processing dei messaggi. Inoltre consente l'interazione delle varie applicazioni con l'utente e con i vari sensori installati sull'auto utili a raccogliere le informazioni di guida.

- Il *Network Layer* ha lo scopo di distribuire i dati alle applicazioni, attraverso specifiche modalità di routing. In particolare vengono definiti tre possibili schemi: *Geographical broadcast*, dove i pacchetti sono distribuiti a tutti i nodi all'interno di una certa area geografica; *Single-hop broadcast*, il quale distribuisce i pacchetti solamente ai nodi vicini aggregando eventualmente le informazioni ad ogni hop; *Beacon packets*, che definisce una particolare forma di Single-hop broadcast e della quale ne discuteremo in seguito.

- Il *MAC Layer* si basa sul protocollo *IEEE 802.11 MAC* adottando come algoritmo di accesso al canale lo standard *Carrier Sense Multiple Access with Collision Avoidance (CSMA/CA)*. Le principali funzionalità a questo livello sono la gestione e il coordinamento dei diversi canali di comunicazione, fornendo ai livelli superiori una stima del carico di essi.

- Il *Physical Layer* anch'esso si basa sul protocollo *IEEE 802.11 PHY*, ma con modifiche per adattarsi al meglio all'ambiente veicolare.

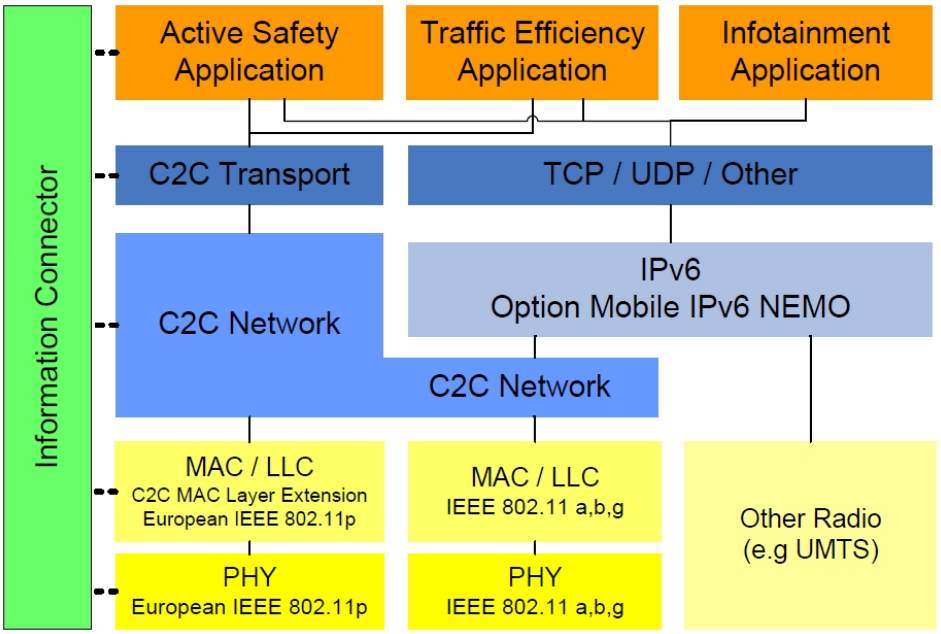

Figura 2.2: Protocolli di comunicazione di una VANET

Un'accurata trattazione degli ultimi due livelli verrà fatta nella sezione successiva, nella quale verrà presentata l'evoluzione dell'IEEE 802.11 adottata per ambienti veicolari: l'*IEEE 802.11p*.

2.2 Il canale radio

Nel 1999 la *Federal Communication Commission (FCC)* ha riservato negli USA sette canali da 10 MHz nella banda dei 5.9 GHz (5.855 - 5.925) denominata *Dedicated Short Range Communication (DSRC) [24]* per uso esclusivo tra Veicolo e Veicolo e Veicolo e Infrastruttura *(V2V e V2I)*.

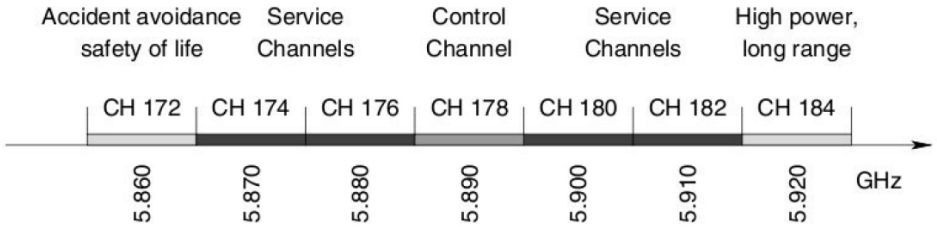

Figura 2.3: La banda DSRC negli USA

Quasi dieci anni dopo, nel 2008, l'*Electronic Communications Committee (ECC)* ha deciso di riservare per l'Europa cinque canali da 10 MHz nella stessa banda (5.875 - 5.925), due dei quali destinati ad un uso futuro [25].

non-safety		Road Safety and traffic efficiency	Control Channel	Critical road safety	Route safety and traffic efficiency	
CH172	CH174	CH176	CH178	CH180	CH182	CH184
5.86	5.87	5.88	5.89	5.90	5.91	5.92 (GHz)
Raccomandata		*Banda designata per ITS*			*Estensione futura*	

Figura 2.4: La banda DSRC in Europa

Come mostrato nelle figure 2.3 e 2.4, il canale 178 è *Control Channel (CCH)* riservato esclusivamente alle comunicazioni di sicurezza. I canali restanti, denominati *Service Channel (SCH)*, sono utilizzati invece per le altre applicazioni. Un dispositivo 802.11 operante nella banda DSRC viene detto in *WAVE mode (Wireless Access in Vehicular Environments)*, formalizzato da IEEE tramite l'emendamento 802.11p [26] e adattato al contesto Europeo nello standard *ITS-G5* [27].

2.2.1 Lo standard IEEE 802.11p per sistemi WAVE

Lo standard per DSRC 802.11p WAVE [28], nasce come aggiustamento ad hoc dello standard 802.11 [29] con lo scopo di ridurre il più possibile le dimensioni dei pacchetti da inviare tramite il collegamento radio. Si occupa di descrivere le funzioni e i servizi richiesti per operare in rapidità, scambiando messaggi senza la necessità di creare una *Basic Service Set (BSS)*, cioè una tipologia di rete wireless in cui è presente un *Access Point (AP)*.

Il livello MAC

Nello strato protocollare MAC di 802.11, il principio di funzionamento di una BSS si basa su una fase iniziale di *handshake* tra le stazioni wireless e l'AP, per stabilire e mantenere una connessione sicura. La BSS è identificata univocamente a livello MAC da un Basic Service Set Identification (BSSID) che corrisponde all'indirizzo MAC dell'AP.

L'idea proposta da 802.11p a questo livello, è quella di creare una comunicazione efficiente senza l'overhead generato dalle operazioni di handshake, troppo onerose per le reti veicolari, a vantaggio della velocità di trasmissione. Per questo motivo è stato pensato di settare tutti i dispositivi sullo stesso canale e configurarli con lo stesso indirizzo MAC (BSSID). Nasce così il concetto di Wave BSS (WBSS): un insieme di dispositivi in modalità Wave che utilizzano lo stesso BSSID. Una WBSS viene inizializzata dal momento in cui viene trasmesso il primo pacchetto (*beacon*) con tutte le informazioni necessarie affinché chi lo riceve possa configurarsi.

Per quanto riguarda l'accesso al canale, viene adottata una politica CSMA/-CA sfruttando l'*Enhanced Distributed Channel Access (EDCA)* già definito per l'*IEEE 802.11e*. L'EDCA aggiunge delle funzionalità per la gestione della qualità del servizio (*QoS*) aggiungendo meccanismi di riscontro (ACK) e la possibilità di trasmettere continuativamente. Inoltre offre quattro diverse code, dette *Channel Access Function (CFA)*, distinte dall'uso di quattro diversi gruppi di parametri di contesa d'accesso. Queste classi di accesso permettono di far coesistere applicazioni di sicurezza, per le quali è indispensabile una tempistica stringente, con gli altri tipi applicazioni (VoIP, Business), meno sensibili a ritardi.

Il livello Fisico

A livello fisico è stata adottata la filosofia di apportare il minor numero di cambiamenti rispetto all'esistente 802.11 PHY. Questo è stato possibile perché *IEEE 802.11a* opera nella frequenza dei 5GHz e non sono state necessarie troppe modifiche per renderlo operativo a 5.9GHz. E' stata utilizzata come modulazione l'*Orthogonal Frequency Division Modulation (OFDM)* con larghezza di banda del canale di 10MHz anziché i 20Mhz di 802.11a. Questa scelta è stata introdotta per ridurre gli effetti del rumore presenti nel canale radio. In figura 2.5 sono mostrate le principali differenze tra i due standard.

	IEEE 802.11a	IEEE 802.11p
Data rate	6, 9, 12, 18, 24, 36, 48, 54 Mbps	3, 4.5, 6, 9, 12, 18, 24, 27 Mbps
Modulation	BPSK OFDM QPSK OFDM 16-QAM OFDM 64-QAM OFDM	BPSK OFDM QPSK OFDM 16-QAM OFDM 64-QAM OFDM
Error Correction Coding	Convolutional Coding with K=7	Convolutional Coding with K=7
Coding Rate	1/2, 2/3, 3/4	1/2, 2/3, 3/4
# of subcarriers	52 net	52 net
OFDM Symbol Duration	4.0 μs	8.0 μs
Guard Period	0.8 μs	1.6 μs
Occupied bandwidth	20 MHz	10 MHz
Frequency	5 GHz ISM band	5.850-5.925 GHz

Figura 2.5: Differenze tra 802.11a e 802.11p

2.3 I protocolli di comunicazione

Mentre lo sviluppo delle applicazioni è ancora in fase di definzione più precisa, la ricerca ha portato ad evidenziare differenti classi di modalità di comunicazione [30]. Possiamo identificarne cinque distinte tra loro: *Beaconing*, *Geobroadcast*, *Routing Unicast*, *Information Dissemination* e *Information Aggregation*. Sulla base di questi metodi di comunicazione potranno essere sviluppate tutte le applicazioni attuali e future. Vediamo molto brevemente una descrizione di queste ultime quattro classi, mentre dato che in questa tesi l'applicativo si basa sul protocollo di Beaconing, nelle sezioni successive andre-

mo ad analizzare dettagliatamente la sua forma e la tipologia di applicazioni basate su di esso.

- *Geobroadcast*: lo scopo di questa classe di applicazioni è quello di distribuire immediamente delle informazioni all'interno di un'ampia area, ad esempio per avvisare un veicolo in arrivo di un determinato evento oppure per segnalare condizioni stradali anomale che necessitano l'attenzione del guidatore. Ogni nodo trasmette in broadcast il messaggio a tutti i suoi vicini, che lo inoltrano solo entro i limiti dell'area designata (figura 2.6). In situazioni di alta densità stradale, per garantire una migliore scalabilità possono essere adottati schemi di inoltro per evitare la troppa ridondanza.

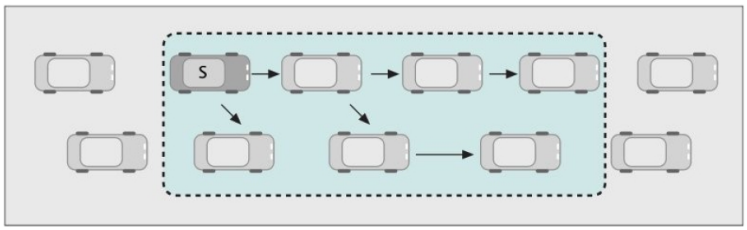

Figura 2.6: Propagazione dei messaggi inviati in modalità Geobroadcast

- *Routing Unicast*: in questo caso la rete veicolare è utilizzata per il trasporto punto-punto di un messaggio (figura 2.7), e non per la distribuzione di messaggi. In base a questi requisiti, è necessario un meccanismo capace di determinare la posizione corrente della destinazione, a partire dalla sua identità. Nelle reti IP cablate l'*Address Resolution Protocol (ARP)* provvede alla determinazione dell'indirizzo fisico; nelle VANET la funzione di ARP è sostituita da un *Location Service* che può essere realizzato in forma *reattiva* o *proattiva*. Nel primo caso una volta raggiunta la destinazione è in grado di conoscere anche l'instradamento. Nel secondo caso invece, rimanda la scoperta del routing ad una fase successiva.

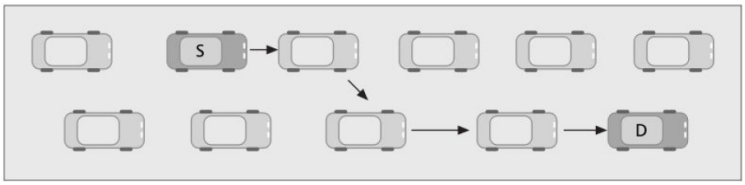

Figura 2.7: Propagazione dei messaggi in modalità Routing Unicast

- *Information Dissemination*: ha lo scopo di mantenere viva un'informazione per prolungati periodi di tempo, rendendola disponibile anche a

chi arriva successivamente (figura 2.8). I messaggi possono essere prioritizzati in base al contesto e al volume del traffico. Lo schema di comunicazione riprende il classico broadcast singlehop, con l'aggiunta di alcuni parametri per capire quando reinviare un messaggio.

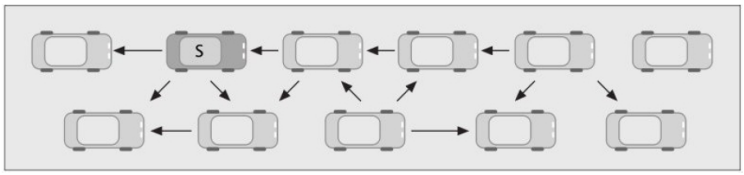

Figura 2.8: Propagazione dei messaggi in modalità Information Dissemination

- *Information Aggregation*: utilizzata per ridurre il sovraccarico della rete. I messagi ricevuti vanno ad arricchire una base di conoscenza del veicolo, a partire dal quale sono emessi nuovi messaggi (figura 2.9). In questo modo al verificarsi di un evento, individuato da più veicoli, è possibile evitare uno storms di messaggi attraverso l'aggregazione dei dati.

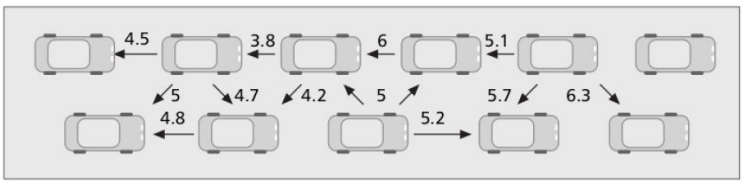

Figura 2.9: Propagazione dei messaggi in modalità Information Aggregation

2.3.1 Il Beaconing

I messaggi di Beaconing (es. in figura 2.10) vengono trasmessi da tutti i veicoli ad intervalli regolari compresi tra 0.1s e 1s. I messaggi devono avere dimensioni ridotte e possono contenere alcune informazioni destinate ai nodi limitrofi tra cui l'identità del veicolo, la posizione, la direzione, la velocità ed eventuali messaggi derivanti da qualche sensore attivo sull'auto. I beacon non sono inoltrati inalterati e concorrono alla consapevolezza cooperativa: ciò significa che prima di inviare nuovi beacon, ogni veicolo elabora i dati ricevuti dai veicoli adiacenti modificando, se necessario, il prossimo messaggio da inviare.

I messaggi sono inviati in modalità broadcast a tutti i veicoli che rientrano nella portata dell'antenna. Di regola l'invio dei beacon è effettuato ad intervalli temporali costanti ma potrebbe accadare che, a causa di un evento particolare (ad es. un incidente), vi sia una cambiamento del periodo dell'intervallo di

invio. Tra tutti i messaggi che possono essere inviati nel canale radio, i beacon hanno la priorità più bassa.

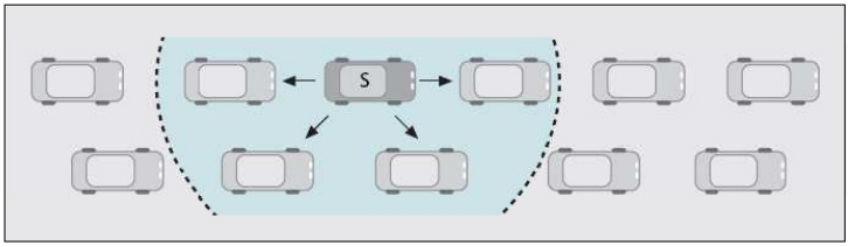

Figura 2.10: Messaggi di beaconing periodici inviati in broadcast

2.3.2 Applicazioni basate su beaconing

Prima di descrivere alcuni tipi di applicazioni che sfruttano il beaconing, è importante sapere quali sono i criteri specifici di comunicazione che distinguono le differenti classi di applicazione. Questi requisiti sono stati definiti dall'*European Telecommunications Standards Institute (ETSI) Technical Committee for ITS (ETSI TC ITS)* [31] e sono:

- *Minimum packet transmission frequency [Hz]*: rappresenta la frequenza minima di trasmissione di pacchetti alla quale i veicoli devono trasmettere le loro informazioni. I possibili valori variano in un range di 1Hz - 10Hz.

- *Maximum latency time [ms]*: denota il tempo di latenza massimo tra la generazione del pacchetto ai livelli alti e l'effettiva trasimssione attraverso il canale wireless. Richiede valori tra 50ms e 500ms.

- *Minimum duration of the total exchange [s]*: dipende dal caso d'uso, dalla velocità e dal range di trasmissione. Non sono richiesti valori specifici.

- *Absolute/relative positioning accuracy [m]*: richiesta quando è necessaria una differenziazione delle corsie. I valori possibili sono tra 1m e 20m.

- *Authentication/security requirements*: è particolarmente richiesto per veicoli di emergenza e operazioni commerciali.

- *Availability of digital map information*: alcune applicazioni richiedono la conoscenza dell'ambiente specifico della strada per operare adeguatamente.

Per quanto riguarda le applicazioni di beaconing, sviluppate per la classe dell'*Active road safety* [32], è di fondamentale importanza la frequenza minima di trasmissione: infatti l'ETSI TC ITS ha stabilito la necessità di inviare 10 pacchetti al secondo, settando quindi tale valore a 10Hz. Di seguito vediamo due possibili casi d'uso.

Intersection collision warning (ICW)

Quest'applicazione appartiene alla sottoclasse delle *Co-operative awareness*. Avvisa il guidatore quando una potenziale collisione ad un incrocio è rilevata attraverso lo scambio dei pacchetti tra i veicoli. Può essere applicata ad incroci con o senza semafori e può essere basata su comunicazioni V2V o V2I.

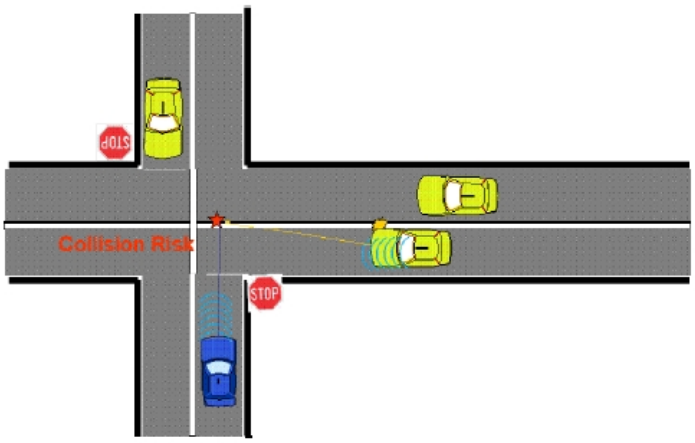

Figura 2.11: Scenario di Intersection collision warning

Nel primo caso, i veicoli inviano periodicamente in broadcast i pacchetti per localizzarsi tra loro prima di raggiungere l'incrocio (figura 2.11). Nel secondo caso, previsto quando la visibilità è ostruita (NLOS), una RSU è installata sull'incrocio col compito di ricevere e processare tutti i pacchetti ricevuti dai veicoli in avvicinamento, rispondendo con una serie di pacchetti di avviso quando individua la possibile collisione tra i due veicoli. Indipendentemente dallo scenario di comunicazione, i veicoli devono necessariamente ricevere le informazioni su una potenziale collisione in tempi sufficienti (o entro una certa distanza), per garantire al guidatore di decelerare ed evitare l'incidente all'incrocio.

Requisiti ICW
Minimum frequency of the periodic message: 10Hz
Maximum latency time: 100ms
Maximum required communication range: 300m
Accurate positioning of vehicles on digital maps

Tabella 2.1: Requisiti di ICW

Emergency electronic brake light (EEBL)

Quest'applicazione appartiene alla sottoclasse delle *Road hazard warning*. Quando un veicolo frena violentemente, l'applicazione manda automaticamente per un certo periodo dei pacchetti di avviso agli altri veicoli vicini per cercare di evitare o comunque limitare un tamponamento (figura 2.12). Quest'applicazione aiuta i veicoli che seguono attraverso una tempestiva notifica della brusca frenata, anche quando la visibilità del guidatore è limitata. In questo caso, il pacchetto trasmesso include informazioni di localizzazione come la posizione della situazione e l'area in cui propagare i messaggi. E' possibile inoltre supportare quest'applicazione con tradizionali tecnologie radar per prevenire ancora di più possibili incidenti.

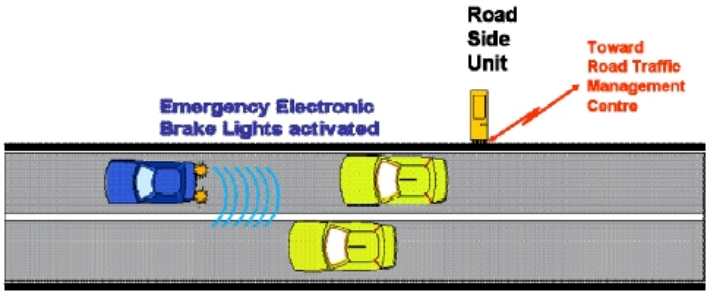

Figura 2.12: Scenario di Emergency electronic brake light

Requisiti EEBL
Minimum frequency of the periodic message: 10Hz
Maximum latency time: 100ms
Maximum required communication range: 300m

Tabella 2.2: Requisiti di EEBL

Capitolo 3

Gli esperimenti

In questo capitolo verranno descritti dettagliatamenti gli esperimenti svolti, focalizzando l'attenzione sul setup fisico e le misure calcolate. Dato che uno degli obiettivi di questa tesi è stato quello di valutare la conformità dei risultati ottenuti con quelli precedentemente calcolati, la scelta delle componenti hardware e software è stata fatta sulla base degli esperimenti passati. La descrizione del software implementato viene rimandata al capitolo successivo.

Gli esperimenti sono stati svolti in due fasi: la prima ha avuto lo scopo di testare il sistema per valutarne la correttezza e di individuare eventuali errori non riscontrabili in laboratorio. Questa fase è stata svolta in un tratto urbano periferico di Pisa. La seconda fase invece è servita alla raccolta dei dati per la realizzazione dei report e si è articolata su tre giorni, con un kilometraggio totale di circa 500km. In seguito saranno presentati solamente i dettagli della seconda fase.

3.1 Il setup fisico

Il setup fisico è composto da tutte le componenti hardware necessarie alla comunicazione dei tre veicoli. La componente principale è l'unità *LinkBird-MX*, che costituisce l'OBU del nostro dominio veicolare; di seguito vedremo le sue caratteristiche principali, per poi introdurre tutti i componenti utilizzati nella realizzazione degli esperimenti.

3.1.1 Le LinkBird-MX

I laboratori NEC hanno realizzato le LinkBird-MX come prototipo per le attività di ricerca mirate allo sviluppo di applicazioni per comunicazioni vei-

colari. Ogni unità è formata da due parti: la piattaforma hardware, visibile in figura 3.1, e le librerie software integrate chiamate *CAR-2-X Software Developement Kit (C2X SDK)* [33], dove la CAR-2-X sta per *CAR-2-CAR* e *CAR-2-INFRASTRUCTURE*.

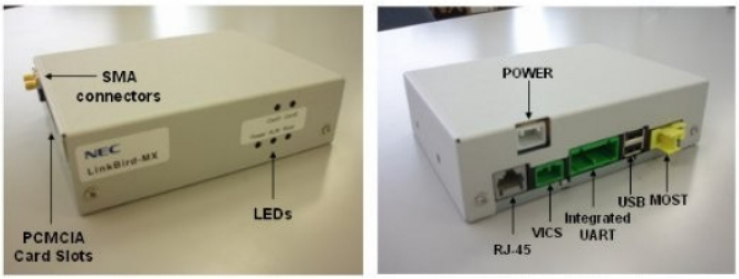

Figura 3.1: Vista frontale e dal dietro di una unità Linkbird

L'hardware attualmente utilizzato ospita un kernel linux alla versione 2.6.19 supportato da una CPU MIPS a 64 bit, 512 MB NAND-Flash, 16 MB Nor-Flash e 128 MB SDRAM. Ha dimensioni piuttosto ridotte (153,5mm x 118mm x 43mm) ideali per l'utilizzo come OBU sul veicolo. Non dispone di una batteria e quindi necessita di essere collegata ad una fonte di corrente: dato che si potrebbe incorrere in problemi di esaurimento della batteria del veicolo, sono state studiate tecnologie per minimizzare il consumo di potenza (al massimo 5W). Può ospitare processi applicativi oppure essere usata a partire da un diverso computer collegato via Ethernet (caso d'uso di questa tesi).

Le LinkBird-MX hanno la possibilità di interfacciarsi via porta Ethernet, 2 USB, GPS, CAN (Controlled Area Network) e MOST (Media Oriented Systems Transport). Dispongono inoltre di una PCMCIA per poter inserire due altre schede WLAN e di due connettori SMA per supportare diversi tipi di antenna. Un riassunto delle varie interfacce è mostrato 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
Connettori SMA	2 porte
PCMCIA	2 slots
Mini-PCI:IEEE802.11p/a/b/g	1 slot
Power(DC12V)	1 porta

Tabella 3.1: Interfacce LinkBird-MX

Il *NEC C2X SDK* è un pacchetto software, scritto in Java, che implementa lo stack protocollare di comunicazione tra veicoli e infrastrutture basato sulla tecnologia wireless a corto raggio. E' formato di due moduli principali: il *C2X Protocol Stack* e le *C2X API*. Il C2X Protocol Stack, mostrato in figura 3.2, implementa le comunicazioni wireless ad-hoc e multi-hop tra OBU e RSU, sfruttando le tipologie di routing precedentemente descritte. Viene eseguito come un demone in user space, comunicante con le applicazioni mediante una API, oppure direttamente per mezzo di Socket UDP. Tra le principali funzionalità implementate, di particolare importanza sono il controllo di potenza per pacchetto, la prioritizzazione dei messaggi, il supporto IPv4 e IPv6 e la gestione della sicurezza tramite firme digitali e certificati.

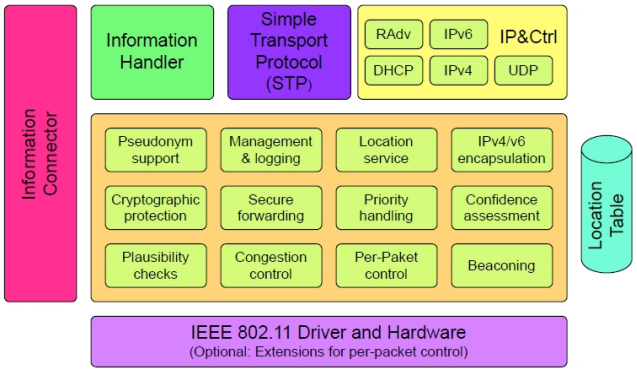

Figura 3.2: C2X Protocol Stack

Le C2X API (Applications Programming Interfaces), mostrate in figura 3.3, mettono a disposizione del programmatore delle interfacce per comunicare con lo Stack Protocollare. Le API permettono di eseguire le applicazioni su unità dedicate (ad esempio un portatile) e grazie alla portabilità di Java, possono essere eseguite su qualsiasi sistema operativo.

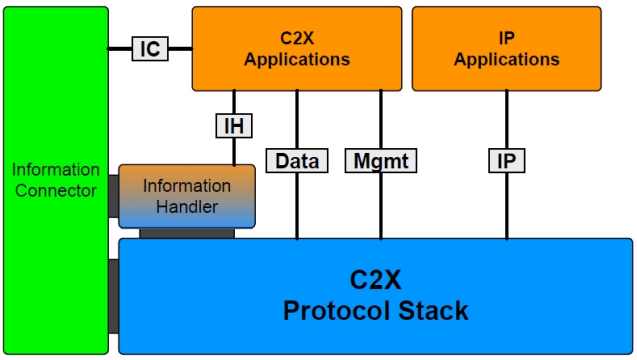

Figura 3.3: C2X API

Vengono distinti due tipi di applicazioni che possono utilizzare le API: *Applicazioni C2X* e *Applicazioni IP*. Le prime si basano sul protocollo veicolare e ne possono controllare meccanismi e funzionalità. Le seconde si basano sul protocollo standard IP e utilizzano la rete veicolare come un canale trasparente. Sulla base di questa distinzione, il C2X SDK mette a disposizione sei tipi di interfacce:

- Data e Management. Permettono di inviare/ricevere messaggi e di leggere/scrivere la posizione geografica, gli pseudonimi e le credenziali di sicurezza, gli elementi della *location table* e configurare un *location service* (tecnica di localizzazione geografica).
- Information Connector. Permette lo scambio asincrono di informazioni tra gli strati, con le applicazioni che si iscrivono a servizi di stato e sono notificate dell'occorrenza di eventi, ad esempio quando appare un nuovo vicino.
- Information Handler. Aggrega le informazioni di più applicazioni e le inserisce nei Beacon.
- IP. Consente l'esecuzione di programmi basati su IPv4/6.
- Network and Transport Access (NWTA). Nuova interfaccia implementata per supportare applicazioni per progetti *Pre-Drive C2X*.

Concludendo, il C2X SDK è stato sviluppato con l'intento di essere indirizzato a due principali gruppi di potenziali utenti: il primo, quello degli sviluppatori di applicativi, può utilizzare le API SDK per uno sviluppo rapido del software, affidandosi ai sottolivelli messi a disposizione dallo stack protocollare dall'SDK stesso. Il secondo, quello degli ingegneri di sistema, può utilizzare l'SDK per condurre espirimenti, testando delle specifiche configurazioni e misurando le performance di particolari funzionalità.

3.1.2 Gli altri componenti

Il setup fisico si completa con i seguenti componenti:

- computer portatile Sony Vaio VGN-Z21MN con processore Intel Core 2 Duo CPU P8600 @ 2.40 Ghz, 4GiB di Memoria Ram, con sistema operativo Ubuntu 11.10 con kernel 3.0.0-19.

- computer portatire Sony Vaio con processore Intel Core 2 Duo CPU P7350 @ 2.00 Ghz, 3GiB di Memoria Ram, con sistema operativo Ubuntu 9.04 con kernel 2.6.28-19.
- computer portatile Acer Travelmate con processore Inter Celeron CPU 530 @ 1,73 Ghz, 1GiB di Memoria Ram, con sistema operativo Ubuntu 9.10 con kernel 2.6.31-22.
- tre antenne abbinate alle LinkBird-MX
- tre cavi Ethernet incrociati per la connessione dei portatili alle LinkBird-MX
- tre cavi Seriali utili al settaggio delle impostazioni delle LinkBird-MX
- tre ricevitori Model-B07 GPS per computer con attacco USB
- tre inverter per trasformare la corrente continua dell'auto in alternata, necessari per fornire energia a tutti i componenti
- veicolo *Lancia Phedra*, altezza 1.74m, utilizzato come veicolo alto
- veicolo *Ford Galaxy*, altezza 1.76m, utilizzato come veicolo alto
- veicolo *Alfa Romeo Giulietta*, altezza 1.47m, utilizzato come veicolo basso
- veicolo *Volkswagen Golf*, altezza 1.48m, utilizzato come veicolo basso
- veicolo *Lancia Delta*, altezza 1.50m, utilizzato come veicolo basso
- veicolo *Fiat Marea*, altezza 1.46m, utilizzato come veicolo basso

La presenza di più auto differenti è stata dettata dalla disponibilità di esse al momento del noleggio. In seguito, per ciascuno dei 3 report verranno riportate quali sono state utilizzate.

Il dominio all'interno dell'auto, mostrato in figura 3.4, comprende quindi la presenza di un portatile connesso tramite USB ad un ricevitore GPS e tramite Ethernet alla LinkBird-MX, a sua volta collegata col connettore SMA all'antenna. Quest'ultima (figura 3.5) è stata posizionata sul tettino dell'auto in una posizione centrale mentre il ricevitore GPS è stato posizionato sul parabrezza dell'auto per garantire una migliore ricezione del segnale (figura 3.6).

Figura 3.4: Dominio all'interno dell'auto: portatile e linkbird-MX

Figura 3.5: Antenna LinkBird-MX

Figura 3.6: Ricevitore Model-B07 GPS

Figura 3.7: Situazione generale dei tre veicoli

In figura 3.7 è mostrata una configurazione delle tre auto per gli esperimenti su strada. Dalla foto è possibile notare la scelta di un auto di differente altezza rispetto alle altre due, in questo caso posizionata in testa ai tre veicoli.

3.2 I report finali

Come presentato negli obiettivi di questa tesi, uno degli scopi primari era quello di valutare le performance di un'applicazione di beaconing in un ambiente veicolare multihop. Prima di definire accuratamente quali sono state le misure considerate nello studio delle performance, bisogna delineare due modelli: singlehop/multihop, LOS/NLOS.

3.2.1 Singlehop/Multihop

Quando si parla di comunicazione multihop, solitamente si intende il processo di comunicazione per lo scambio di informazioni tra due nodi di una rete, non collegati direttamente tra loro. Si ha quindi la possibilità di far viaggiare inalterato un pacchetto nella rete per un numero (limitato) di nodi, calcolando il tempo di attraversamento del pacchetto dal nodo mittente al nodo destinatario. Il concetto di beaconing richiede però specifiche diverse: ogni nodo della rete contribuisce alla conoscenza cooperativa, inviando ad intervalli regolari le informazioni più aggiornate di tutti i suoi nodi vicini. In questo scenario cambia sensibilmente il concetto di multihop: si tratta infatti della possibilità per un veicolo x, comunicante con y, di ricevere informazioni più recenti relative a y stesso da un terzo veicolo z.

Ad esempio considerando lo scenario in figura 3.8, analizziamo questa situazione: al tempo t il veicolo 0 invia in broadcast un pacchetto attraverso i canali $SH(0,1)$ e $SH(0,2)$. Al tempo $t+1$ supponiamo che il veicolo 1 riceva tale pacchetto e il veicolo 2 no, a causa della maggiore distanza e la presenza di un ostacolo. Il veicolo 1 dopo aver aggiornato la propria tabella, invia un nuovo pacchetto contenente anche le informazioni sul veicolo 0 appena aggiornate sul canale $SH(1,2)$. Al tempo $t+2$ il veicolo 2 riceverà il pacchetto con le informazioni più aggiornate sul veicolo 0 dal veicolo 1, andando a sua volta ad aggiornare la propria tabella. Sulla base di questo a noi interessa per ogni misura calcolata, valutare il beneficio apportato dalla comunicazione multihop, in cui le informazioni relative al veicolo 0 ricevute dal veicolo 1 sono prese in considerazione, rispetto a quella singlehop, in cui invece tali informazioin sono ignorate.

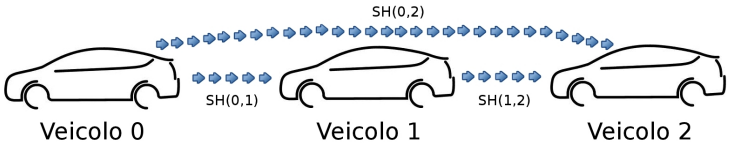

Figura 3.8: Scenario di comunicazione multihop

3.2.2 LineOfSight (LOS)/NonLineOfSight (NLOS)

Questo modello, identificato come L/N, differenzia il canale comunicativo singlehop tra due veicoli in termini di visibilità: nel caso LOS la comunicazione è diretta, senza alcun ostacolo che si interpone tra i due. Invece nel caso NLOS, la comunicazione è ostacolata da un oggetto, come ad esempio un'altra automobile, un camion oppure un muro in prossimità di un incrocio.

A differenza dei precedenti esperimenti, svolti con solo due veicoli, è stato possibile caratterizzare questo modello misurando le differenze di performance tra il caso LOS e NLOS. In particolare oltre a darne una panoramica generale, sono state studiate delle configurazioni in base ai seguenti parametri:

1. il tipo di altezza dei due veicoli comunicanti in modalità LOS: è possibile quindi distinguere in basso/basso, alto/basso, basso/alto. (il primo veicolo è il trasmittente, l'altro il ricevente)

2. il tipo di altezza del veicolo posto come ostacolo alla comunicazione degli altri due: in questo caso si distingue in basso e alto.

3.2.3 Le misure calcolate

Tutte le misure di seguito elencate, oltre ad essere state calcolate sull'aggregazione generale dei dati, sono state definite più specificatamente sulla base dei due modelli sopra definiti.

- *Beacon (packet) delivery rate (PDR)*: si riferisce al rapporto tra i beacons correttamente ricevuti e il numero totale di beacons trasmessi, nella comunicazione diretta tra due veicoli. Nel caso multihop vanno sommati al numeratore, i beacons ricevuti da altri veicoli.

- *PDR vs. Distance*: da una stima del PDR sulla base della distanza. E' utile per capire fino a che distanza i beacons possono essere ricevuti con un minimo di affidabilità.

- *Beacon (packet) inter-reception (PIR) time*: è definito come l'intervallo di tempo trascorso tra la ricezione di due beacons consecutivi. E' la misura principale del lavoro svolto, in quanto mostrerà se i requisiti minimi definiti per le applicazioni di beaconing sono soddisfatti.

- *PIR BlackOut*: indica la probabilità che l'intervallo di PIR superi 1 secondo, definito come tempo di balckout. Dato che un'automobile può percorrere fino a 30-40 m al secondo in autostrada, questa misura ci mostra qual è la probabilita che potenziali situazioni pericolose possano non essere rilevate, a causa dell'assenza di informazioni aggiornate.

Capitolo 4

Il software realizzato

In questo capitolo verrà descritto il codice implementato e utilizzato negli esperimenti. Saranno mostrate le scelte implementative allegando dove necessario parti di codice. Sono stati sviluppati due tipi di programmi: l'applicazione di beaconing, utilizzata per la raccolta dei dati, e il programma di post-processing, necessario per redigere i report dei capitoli successivi.

4.1 L'applicazione di beaconing

L'applicazione di beaconing costituisce l'AU all'interno del dominio del veicolo, che eseguita sul portatile comunica con l'OBU. E' stata sviluppata in Java, utilizzando l'ambiente di sviluppo Eclipse, sfruttando le librerie C2X SDK.

Lo sviluppo dell'applicazione ha previsto sostanziali modifiche sulla base di quanto già era stato fatto: in ordine di importanza, la necessità di utilizzare una tabella per memorizzare le informazioni dei veicoli, le modalità di compressione dei dati all'interno del pacchetto e infine la struttura di salvataggio degli invii e delle ricezioni all'interno dei file di log.

Con l'introduzione di un terzo veicolo e in ottica di utilizzo anche con più veicoli, è stato necessario introdurre un'opportuna struttura dati necessaria alla memorizzazione delle informazioni ricevute sugli altri veicoli. Questa scelta è stata dettata dal fatto che un requisito fondamentale delle applicazioni di beaconing nel campo della sicurezza attiva, è quello che ciascun veicolo contribuisce alla conoscenza cooperativa e alla distribuzione in broadcast di tutte le informazioni utili ad altri veicoli per comprendere al meglio lo scenario stradale circostante. Nonostante la semplicità dei test previsti per questa tesi (scenario con solo tre auto) che poteva prevedere l'utilizzo di una struttura dati non

complessa, come un array o un vettore, è stato fatto uno studio accurato che ha portato a scegliere una struttura idonea anche per scenari più complessi.

Il requisito principale della struttura da utilizzare, era quello di dover accedere ad essa in maniera veloce tramite gli ID dei veicoli, permettendo così un rapido aggiornamento delle informazioni. Per questa ragione è stato optato di utilizzare una forma di *mappa*, cioè una collezione di oggetti il cui scopo principale è quello di rendere veloci ed efficienti operazioni quali inserimento e ricerca di elementi. Per fare questo memorizza coppie *<chiave, valore>* e ha due implementazioni disponibili nelle API Java dell'interfaccia *java.util.Map*: *HashMap* e *TreeMap*. Dato che non era di nostro interesse mantenere un ordine nell'inserimento degli elementi nella struttura (funzionalità offerta dalla TreeMap), è stato scelto di utilizare la HashMap anche per le migliori performance delle operazioni di *get* e *put*.

Inoltre abbiamo ritenuto interessante la possibilità di impostare due fattori che, in casi stradali complessi, giocano a nostro parere un ruolo fondamentale nelle performance della tabella: la *capacità iniziale* e il *fattore di carico*. Il primo identifica la capacità al momento in cui la tabella hash viene creata mentre il secondo è la misura di quanto deve essere riempita la tabella affinché la sua capacità venga automaticamente aumentata. Se a questi due parametri aggiungiamo una funzione di refresh in grado di rimuovere le entry non viste per un certo intervallo di tempo, possiamo mantenere una struttura le cui performance non incidano negativamente sulla comunicazione. Tutte queste considerazioni sono state fatte per possibili utilizzi futuri in quanto ai fini dei nostri esperimenti risultavano inutili. Infatti la capacità iniziale è stata impostata ad un valore di 3 mentre il fattore di carico è stato lasciato quello di default (.75); la funzione di refresh invece non è stata implementata.

Nel listato 4.1 sono mostrate la dichiarazione della HashMap e l'inserimento e la modifica di una entry all'interno di essa.

```
/* Inizializzazione HashMap */
Map<Integer, EntryTable> tableNode = new HashMap<Integer,
    EntryTable>(3);

/* Inserimento ID veicolo 0 */
tableNode.put(Integer.valueOf(0), new EntryTable());

/* Aggiornamento valori per veicolo con ID = key */
tableNode.get(Integer.valueOf(key)).setParams(pkt, currLat,
    currLong, currSpeed, currHead, currTime);
```

Listing 4.1: Inizializzazione e modifica Hash Map

La HashMap come abbiamo detto memorizza coppie *<chiave, valore>* che nel nostro caso sono formate da *<ID veicolo, ultima informazione ricevuta>*. Poiché l'ultima informazione ricevuta è formata da un insieme di valori (latitudine, longitudine, ecc..), è stata creata una classe unica per la memorizzazione di queste informazioni, come mostrato nel listato 4.2.

```
public class EntryTable {

  private int id_Pkt;
  private float latitude;
  private float longitude;
  private float speed;
  private float heading;
  private long time;

  /* Costruttore per prima inizializzazione */
  public EntryTable(){
    id_Pkt = 0;
    latitude = 0;
    longitude = 0;
    speed = 0;
    heading = 0;
    time = 0;
  }

  /* Costruttore con inzializzazione dei parametri */
  public EntryTable(int idPkt, float latitude, float longitude,
      float speed, float heading, long time){
    this.id_Pkt = idPkt;
    this.latitude = latitude;
    this.longitude = longitude;
    this.speed = speed;
    this.heading = heading;
    this.time = time;
  }

  /* Metodo per la modifica dei valori */
  public void setParams(int idPkt, float latitude, float longitude
      , float speed, float heading, long time){
    this.id_Pkt = idPkt;
```

```
    this.latitude = latitude;
    this.longitude = longitude;
    this.speed = speed;
    this.heading = heading;
    this.time = time;
  }

  public int getId_Pkt() {
    return id_Pkt;
  }

  public float getLatitude() {
    return latitude;
  }

  public float getLongitude() {
    return longitude;
  }

  public float getHeading() {
    return heading;
  }

  public float getSpeed() {
    return speed;
  }

  public long getTime() {
    return time;
  }

  public void setHeading(float heading) {
    this.heading = heading;
  }

  public void setId_Pkt(int id_Pkt) {
    this.id_Pkt = id_Pkt;
  }

  public void setLatitude(float latitude) {
    this.latitude = latitude;
  }

  public void setLongitude(float longitude) {
    this.longitude = longitude;
```

```
  }

  public void setSpeed(float speed) {
    this.speed = speed;
  }

  public void setTime(long time) {
    this.time = time;
  }
}
```

Listing 4.2: Classe EntryTable

Infine è stato necessario gestire la sincronizzazione della HashMap: questo per evitare che si creino incosistenze a seguito di operazioni concorrenti di lettura e scrittura presenti nei thread di invio e ricezione di un messagio, discussi in seguito. Dato che la classe HashMap risulta essere non sincronizzata, è stato necessario utilizzare una classe di supporto per gestire la sincroninizzazione esterna. La scelta è ricaduta sulla classe *Semaphore*, che definisce un semaforo su cui effettuare le operazioni di acquisizione e rilascio, come mostrato nel listato 4.3.

```
/*Inizializzaione del semaforo*/
Semaphore sem = new Semaphore(1, true);

/*Acquisizione e rilascio*/
sem.acquire();

{blocco da eseguire in mutua esclusione}

sem.release();
```

Listing 4.3: Gestione del semaforo

Il secondo parametro del costruttore rappresenta la gestione del *fairness*: in questo caso, viene esplicitamente richiesto che la coda con tutti i thread che richiedono l'accesso alle risorse, venga processata secondo il metodo *First In First Out (FIFO)*.

Il successivo problema ha riguardato la modalità di compressione dei dati all'interno dei pacchetti inviati: all'aumentare dei veicoli e, di conseguenza, delle informazioni da inviare, diventa importante scegliere una forma di inse-

rimento dei dati nel pacchetto, senza che questo diventi di dimensioni troppo grandi che potrebbero creare latenze nella trasmissione. Inizialmente erano state adottate possibili soluzioni messe a disposizioni dalle API di Java, come ad esempio la *Serializzazione*; ma causa dell'enorme quantità di dati aggiunta dal processo di serializzazione, è stato scelto di effettuare una trasformazione *manuale*, come mostrato nel listato 4.4.

```
/*COSTRUZIONE PAYLOAD PACCHETTO */

/*aggiorno la tabella nella entru del veicolo mittente*/
tableNode.get(Integer.valueOf(veicolo)).setParams(count2, latitude
    , longitude, speed, heading, time);
byte[] payload = new byte[pkt_size];
int i;

/*veicolo mittente*/
byte veicolo_mittente = (byte) veicolo;
payload[0] = veicolo_mittente;

/*dimensioni tabella*/
byte tablesize = (byte) tableNode.size();
payload[1] = tablesize;


int j = 0, k = 0;
for(Map.Entry<Integer, EntryTable> t : tableNode.entrySet()){
  /*ID veicolo*/
  byte id_veicolo = (byte) t.getKey().intValue();
  payload[2+j] = id_veicolo;

  /*ID pacchetto*/
  byte[] ID = Helpers.toByta((short)t.getValue().getId_Pkt());
  for(i=0;i<ID.length;i++)
    payload[3+i+j]=ID[i];

  /*Latitudine*/
  byte[] latb = Helpers.toByta(t.getValue().getLatitude());
  payload[5+j]=latb[0];
  payload[6+j]=latb[1];
  payload[7+j]=latb[2];
  payload[8+j]=latb[3];

  /*Longitudine*/
  byte[] lonb = Helpers.toByta(t.getValue().getLongitude());
```

```
payload[9+j]=lonb[0];
payload[10+j]=lonb[1];
payload[11+j]=lonb[2];
payload[12+j]=lonb[3];

/*Velocita'*/
byte[] speedb = Helpers.toByta(t.getValue().getSpeed());
payload[13+j] = speedb[0];
payload[14+j] = speedb[1];
payload[15+j] = speedb[2];
payload[16+j] = speedb[3];

/*Direzione*/
byte[] headb = Helpers.toByta(t.getValue().getHeading());
payload[17+j] = headb[0];
payload[18+j] = headb[1];
payload[19+j] = headb[2];
payload[20+j] = headb[3];

/*Tempo GPS*/
byte[] timeb = Helpers.toByta(t.getValue().getTime());
payload[21+j] = timeb[0];
payload[22+j] = timeb[1];
payload[23+j] = timeb[2];
payload[24+j] = timeb[3];
payload[25+j] = timeb[4];
payload[26+j] = timeb[5];
payload[27+j] = timeb[6];
payload[28+j] = timeb[7];

/*Incremento k per ogni veicolo*/
k++;

/*Mi dice a che punto del pacchetto sono arrivato a scrivere*/
j = k*27;

/*Stringa da memorizzare nel file di log*/
s2 += t.getValue().getLatitude()+
 " "+t.getValue().getLongitude()+
 " "+t.getValue().getSpeed()+
 " "+t.getValue().getHeading()+
 " "+t.getValue().getTime()+ " " + t.getValue().getId_Pkt() + "
     ";
}
```

```
/*Byte fittizzi*/
for(i=j;i<pkt_size;i++)
  payload[i]=(byte)i;


/*PARSING PAYLOAD PACCHETTO */

/*veicolo che ha invia il messaggio*/
tmp =Integer.toString(Helpers.toUnsignedInt(payload2[0]));
s+=tmp + " ";

/*variabile che indica la grandezza della tabella*/
int size = Helpers.toUnsignedInt(payload2[1]);

/*variabile utile all'avanzamento nel payload e nella tabella*/
int j = 0, k = 0;

while(k < size){
  /*Id del veicolo di cui seguiranno i dati*/
  int key = Helpers.toUnsignedInt(payload2[2+j]);

  /*Prendo la entry della tabella relativa all'id di valore key*/
  EntryTable et = tableNode.get(Integer.valueOf(key));

  /*Se il campo che leggo e' relativo a me stesso, salto.*/
  if(key != veicolo){

    /*ID pacchetto*/
    byte[] idPkt = new byte[2];
    idPkt[0] = payload2[3+j];
    idPkt[1] = payload2[4+j];
    int pkt = Helpers.toShort(idPkt);

    byte[] llb = new byte[4];

    /*latitudine*/
    llb[0] = payload2[5+j];
    llb[1] = payload2[6+j];
    llb[2] = payload2[7+j];
    llb[3] = payload2[8+j];
    float currLat = Helpers.toFloat(llb);

    /*longitudine*/
    llb[0] = payload2[9+j];
    llb[1] = payload2[10+j];
```

```
    llb[2] = payload2[11+j];
    llb[3] = payload2[12+j];
    float currLong = Helpers.toFloat(llb);

    /*velocita'*/
    llb[0] = payload2[13+j];
    llb[1] = payload2[14+j];
    llb[2] = payload2[15+j];
    llb[3] = payload2[16+j];
    float currSpeed = Helpers.toFloat(llb);

    /*heading*/
    llb[0] = payload2[17+j];
    llb[1] = payload2[18+j];
    llb[2] = payload2[19+j];
    llb[3] = payload2[20+j];
    float currHead = Helpers.toFloat(llb);

    /*time*/
    byte[] timeb = new byte[8];
    timeb[0] = payload2[21+j];
    timeb[1] = payload2[22+j];
    timeb[2] = payload2[23+j];
    timeb[3] = payload2[24+j];
    timeb[4] = payload2[25+j];
    timeb[5] = payload2[26+j];
    timeb[6] = payload2[27+j];
    timeb[7] = payload2[28+j];
    long currTime = Helpers.toLong(timeb);

    /*Se l'id del pkt e' maggiore di quello che ho nella tabella,
       faccio l'aggiornamento*/
    if(pkt > et.getId_Pkt() || pkt == 0){
      tableNode.get(Integer.valueOf(key)).setParams(pkt, currLat,
          currLong, currSpeed, currHead, currTime);
      s += currLat + " " + currLong + " " + currSpeed + " " +
          currHead+
        " " + currTime + " " + pkt + " ";
    }
    else
      s += et.getLatitude() + " " + et.getLongitude() + " " + et.
          getSpeed() + " " + et.getHeading()+
        " " + et.getTime() + " " + et.getId_Pkt() + " ";
  }
  else
```

```
    s += et.getLatitude() + " " + et.getLongitude() + " " + et.
        getSpeed() + " " + et.getHeading()+
      " " + et.getTime() + " " + et.getId_Pkt() + " ";
  k++;
  j = k*27;
}
```

Listing 4.4: Costruzione e parsing del payload pacchetto

L'inserimento viene effettuato attraverso una trasformazione diretta dei dati in un array di byte, che nel nostro scenario ha permesso di ottenere una dimensione massima del pacchetto inferiore ai 150 byte (contro gli oltre 400 byte risultanti dalla serializzazione). In ricezione i dati vengono ricostituiti specularmente (listato 4.4) avendo a disposizione nel pacchetto la dimensione della tabella. Grazie all'HashMap non è necessario che i veicoli abbiano memorizzati i veicoli nel solito ordine, in quanto l'accesso ad essa avviene tramite la chiave rappresentata dagli ID dei veicoli. La classe *Helpers* contiene metodi di supporto ad esempio per la conversione dei tipi, non definite in Java. Nel codice 4.5 è mostrato l'esempio di conversione da *float* ad *array di byte*.

```
/*Metodo per la conversione di un float in un array di byte*/
public static byte[] toByta(float data) {
  return toByta(Float.floatToRawIntBits(data));
}

public static byte[] toByta(float[] data) {
  if (data == null) return null;
  byte[] byts = new byte[data.length * 4];
  for (int i = 0; i < data.length; i++)
    System.arraycopy(toByta(data[i]), 0, byts, i * 4, 4);
  return byts;
}
```

Listing 4.5: Esempio conversione dato primitivo

L'array di byte ottenuto, viene inserito nel messaggio ed inviato attraverso le seguenti chiamate.

```
/*Formo il messaggio*/
final SHBMessage msg = new SHBMessage(
  mgmtIPv4Addr.getHostAddress(), DEFAULT_C2X_PORT, (short) 1,
      PRIORITY,
```

```
  DEFAULT_SHB_APP_PORT, DEFAULT_SHB_APP_PORT,
  false, false, false, send_data);
msg.setSecHdr(false);

byte [] daspedire = msg.assemble();

/Invoco il thread per l'invio*/
shbSRThread.send(daspedire);
```

Listing 4.6: Costruzione ed invio del messaggio

Sia il messaggio di tipo *single-hop broadcast (SHB)* che il thread di invio sono implementati nella libreria C2X. La variabile *send_data* rappresenta il *payload* del messaggio, formata proprio dai dati dei veicoli.

L'ultima modifica ha riguardato la struttura dei due file di log, uno adibito per l'invio e l'altro per la ricezione, utili successivamente nella fase di post-processing. La scrittura dei file è effettuata tramite la memorizzazione dei vari campi all'interno di una stringa scritta ad intervalli regolari. Vediamo un esempio di una riga di file di log, sia di invio che di ricezione, risultanti da un esperimento effettuato (per comodità di lettura la riga è stata spezzata):

```
Riga del file di LOG di Invio
1331806169170
43.697598 10.834174 25.32 58.97 1331806102 15913
43.697483 10.83395 25.979 59.6033 1331806103 15533
43.6974 10.833782 26.85 61.61 1331806102 13437

Riga del file di LOG di Ricezione
1331805231044 0
43.633804 10.575804 25.31 72.58 1331805179 6684
43.63369 10.575204 24.66 70.8349 1331805180 6304
43.633587 10.574756 25.59 70.77 1331805179 4209
```

I due file sono strutturati similarmente se non per la presenza di un campo in più nel file di Ricezione, riguardante l'ID del veicolo mittente (in questo caso il veicolo con ID 0). Per il resto contengono nell'ordine il tempo macchina, accurato al millisecondo, e i dati dei tre veicoli in ordine di ID. Da precisare che mentre nel file di invio c'è corrispondenza biunivoca col pacchetto inviato, in quello di ricezione si scrivono le informazioni più aggiornate, che possono non coincidere con il contenuto dell'ultimo pacchetto ricevuto. Questo controllo è

effettuato nella fase di parsing del payload, verificando che l'ID del pacchetto ricevuto sia maggiore del precedente ID memorizzato nella HashMap.

Il programma permette di definire i seguenti parametri:

- *Id Veicolo* [int]: id univoco che identifica ciascun veicolo.
- *Packet size* [byte]: dimensione massima del pacchetto da inviare.
- *Intervallo* [ms]: tempo che intercorre tra l'invio dei pacchetti.
- *Durata* [s]: intervallo di tempo tra una scrittura e un'altra sui file di log.

Ai fini degli esperimenti, di particolare rilevanza sono i parametri riguardanti la dimensione massima dei pacchetti e l'intervallo di trasmissione. Il primo è stato impostato a 150 byte mentre l'intervallo di trasmissione, settato a 100 ms (10 MHz), è stata una scelta imposta dall'ETSI TC ITS per le applicazioni (di beaconing) per il campo della sicurezza attiva.

E' possibile dividere la struttura del programma in tre parti: l'invio, la ricezione dei messaggi e la lettura dei dati dal ricevitore GPS. L'interrogazione dei dati del GPS avviene attraverso l'utilizzo di un Timer che ogni 100 millisecondi provvede alla lettura e alla gestione delle informazioni. Se il GPS è connesso ad uno o più satelliti, la risposta conterrà dati consistenti e buoni da utilizzare; in particolare di maggior importanza sono stati considerati:

- *Latitudine*: viene misurata in gradi sessadecimali ed indica la latitudine del veicolo rilevata durante l'ultima acquisizione dei dati.
- *Longitudine*: viene misurata in gradi sessadecimali ed indica la longitudine del veicolo rilevata durante l'ultima acquisizione dei dati.
- *Velocità*: viene misurata in metri al secondo ed indica la velocità del veicolo rilevata durante l'ultima acquisizione dei dati.
- *Direzione*: viene misurata in gradi decimali ed indica l'angolo della direzione rispetto al Nord rilevato durante l'ultima acquisizione dei dati.
- *Tempo*: viene misurato in secondi ed indica il tempo corrente rilevato durante l'ultima acquisizione dei dati.

Dato che il dispositivo GPS aggiorna i dati ogni secondo, circa dieci letture risulteranno essere uguali.

Il thread di invio ha il compito di costruire ad hoc il pacchetto da inviare sulla rete ad intervalli regolari. Le informazioni utili da propagare nella rete saranno l'insieme dei dati letti dal dispositivo GPS e l'ID unico del pacchetto generato, valore ad ogni invio sempre crescente. Ogni pacchetto inviato conterrà oltre ai dati del veicolo mittente, anche le informazioni aggiornate di tutti gli altri veicoli, concorrendo così alla conoscenza cooperativa ed alla propagazione multihop dell'informazione. Il thread di ricezione è completamente speculare a quello di invio e ha il compito di effettuare il parsing del pacchetto, andando ad aggiornare eventualmente le informazioni degli altri veicoli sulla base di un confronto dell'ID dell'ultimo pacchetto ricevuto.

4.2 Il programma di Post-Processing

Il programma di Post-Processing costituisce una parte fondamentale di questa tesi, poiché si occupa di analizzare i file di log, generati da uno stesso espirimento, e di restituire dei report specifici a seconda delle misure da calcolare. E' necessario dare una descrizione del lavoro svolto da questo programma, per capire come sia delicato gestire una così enorme mole di dati. E' stato scritto nel linguaggio C.

Bisogna partire dal presupposto che per poter trarre conclusioni su possibili comportamenti dell'ambiente veicolare, è fondamentale che i dati raccolti nei file di log, siano ripuliti dalle inconsistenze generate da un'incorretta lettura dei dati dal ricevitore GPS, dovuta magari ad una perdita di collegamento con i satelliti. Per questo motivo è necessario definire il concetto di posizione *valida*, la quale al tempo t deve soddisfare i seguenti requisiti:

(i). deve essere aggiornata rispetto alla lettura al tempo t - 1;

(ii). la distanza tra le posiziona del veicoli al tempo t e $t-1$ deve essere compatibile con la la velocità letta col GPS, ad esempio

$$d(t-1,t) \leq 1.5 \cdot v_{max}$$

dove $d(t-1,t)$ è la distanza tra la posizioni del veicolo al tempo t e al tempo $t-1$ e v_{max} è la velocità maggiore letta ai due tempi.

Il lavoro di pulizia delle posizioni non valide, richiede però a monte un'operazione di allineamento temporale dell'insieme delle tracce generate dai veicoli. Quest'operazione si basa sul tempo letto dal GPS, che nonostante sia accurato

solo al secondo (in autostrada i veicoli viaggiano oltre i 30m al secondo), viene letto dall'applicazione di beaconing ogni 100ms, garantendo così una precisione al decimo di secondo, valore sufficiente per questi esperimenti. In particolare l'allineamento delle tracce di trasmissione ci permette di calcolare la distanza dei veicoli durante il viaggio. Dopo aver quindi allineato temporalmente le tracce dei veicoli, è possibile identificare dei segmenti di file *validi*, cioè che contengono al loro interno dati consistenti di una lunghezza minima temporale di 30s. In questo modo siamo sicuri che in quella porzione di strada è possibile stabilire con certezza la posizione dei veicoli.

Commentiamo ora alcune parti importanti del programma di post-processing riguardanti l'identificazione delle posizioni valide, il calcolo del valore di PIR e la formazione dei bin per tutti i valori di PIR ottenuti. Partiamo col dire che il programma viene eseguito su una coppia di veicoli A e B, passando come parametri i due file di log di A, il file di log di invio di B, la modalità di processing (singlehop o multihop) e la dimensione dei bin relativi alla distanza tra i due veicoli. Inizialmente vengono aperti i file di log di invio di A e di B e memorizzati in due array. Successivamente si identificano il tempo minimo e massimo comune, utile per l'allineamento delle tracce, e si stampano alcune statistiche come il tempo totale comune e il numero di pacchetti inviati. Segue la parte di identificazione delle posizioni valide come mostrato nel listato 4.7, in cui se la posizione soddisfa la condizione di validità, viene considerata come buona attraverso l'utilizzo di un array di booleani.

```
// questa funzione, data una traccia D lunga Len
// determina quali sono i record che sembrano validi
// e memorizza un flag corrispondente in Flg (settato
// inizialmente a zero con la calloc)

void Fill(sData *D, int Len, unsigned char *Flg){
  int k=0;
  unsigned long CurrT;

  // avanzo nel tempo fino al primo momento
  // in cui le tracce sono sovrapposte
  while (k<Len && D[k].T<Tmin) k++;

  //tempo corrente
  CurrT = D[k].T;

  // ripeto per tutti i record contenuti
```

```
  // nel tempo comune alle due tracce (Tmin<=T<Tmax)
  for (k; k<Len && D[k].T<Tmax; k++){
      // se il tempo letto e' minore o uguale a
      // quello corrente qualcosa non va, quindi proseguo
      if (D[k].T<=CurrT) continue;

      // se sono qui il tempo D[k].T e' maggiore del
      // ultimo D[k-1].T = CurrT considerato

      // Se il salto e' di 1 secondo il recordo promette bene, lo
         analizzo
      if (D[k].T==CurrT+1){
    // Qui ho riscontrato un salto di tempo
    // provo a calcolare la distanza tra le posizioni
    double d,v;
    // distanza tra posizioni
    d = ArcDist(&D[k].pos,&D[k-1].pos);
    // velocita' massima
    v = MAX(D[k].pos.vel, D[k-1].pos.vel);
    v = MAX(5.0,1.5*v);
    // se la distanza percorsa e' compatibile
    // con la velocita' riscontrata, allora
    // mi segno che il record e' buono
          if (d<v)
            Flg[CurrT-Tmin]=1;
  }
      // in ogni caso (salto di 1 o piu' secondi)
      // mi segno l'ultimo tempo letto
      CurrT=D[k].T;
    }
}
```

Listing 4.7: Identificazione delle posizioni valide

L'operazione viene svolta su ambedue gli array e sulla base di questi effettuiamo l'operazione di allineamento identificando i frammenti comuni alle due tracce, col vincolo che i frammenti siano lunghi almeno 30s. Successivamente apriamo il file di log di ricezione di A e andiamo a segnare come ricevuti i pacchetti inviati presenti nell'array di invio di B. A seconda della modalità di processing è possibile includere i pacchetti ricevuti non direttamente dal veicolo B, grazie alla presenza del campo che identifica il veicolo mittente del messaggio, come mostrato nella parte sulla generazione del file di log.

A questo punto è possibile determinare la quantità di pacchetti ricevuti sia

in termini di percentuale rispetto al totale inviati che in funzione della distanza tra i due veicoli. Il primo valore viene calcolato come rapporto tra la quantità di pacchetti ricevuti su il totale dei pacchetti inviati. Il secondo invece utilizza dei bin delle dimensioni passate come parametro del programma: a questo punto per ogni pacchetto ricevuto ad una certa distanza x, verrà incrementato il contatore relativo al bin che contiene questa distanza. Nel calcolare infine i valori di PIR prendiamo l'array dei segmenti buoni, e per ciascun segmento da utilizzare, ad ogni istante di tempo, controlliamo se il pacchetto è stato ricevuto entro una certa distanza e in caso positivo si calcola la differenza tra l'ultimo pacchetto ricevuto e l'attuale. Tutti i valori risultanti vengono scritti in un file passato poi alla funzione *pirbin* che forma dei bin con intervallo passato come parametro. Nei risultati mostrati in seguito, l'intervallo è stato impostato a 100ms e tutti i valori di PIR superiori a 10s sono stati inglobati in un unico bin.

```
// funzione per il calcolo del tempo di PIR
void PIR(double CritDist, char *fileOut){
  int k;
  FILE *f;

  // apro il file dove scrivero' tutti i tempi
  f = fopen(fileOut,"w");
  if (!f){
      fprintf(stderr,"impossibile aprire il file [%s]\n",
       fileOut);
      exit(1);
    }
  // ciclo su tutti i frammenti che sono stati
  // selezionati
  int rec = 0, tot = 0;
  for (k=0; k<nFram; k++)
    if (Frams[k].usa){
  rec += Frams[k].nRec;
      tot += Frams[k].nTot;
  int j,ia1, iz1;
  // non ho ancora visto una ricezione
  int Seen=NO;
  // contatore messaggi che intercorrono
  // tra due ricezioni
  int first = 1, lastRec = 0;
  int Pir;
```

```
   // tempo di inizio e fine del frammento
  ia1=Frams[k].ia1;
  iz1=Frams[k].iz1;

  // ciclo su tutti gli istanti del frammento
  for (j=ia1; j<=iz1; j++){
    double d;
    int b;
    // se le due macchine sono ad una
    // distanza maggiore della soglia critica
    // assumo che il dato non sia significativo,
    // quindi mi metto nella condizione di non
    // aver ancora visto una ricezione e continuo
    if (Snd1[j].dist > CritDist || !Snd1[j].received)
              continue;
    if(first){
      first = 0;
      lastRec = j;
    }
    else{
    Pir = Snd1[j].timeStamp - Snd1[lastRec].timeStamp;
    fprintf(f, "%d\n", Pir);
    lastRec = j;
    }
  }
  }
  printf("Rec %d / %d\n", rec, tot);
  // chiudo il file dei pir
  fclose(f);
}
```

Listing 4.8: Calcolo dei valori di PIR

```
// funzione per la creazione di bin di dimensioni size
void pirbin(int n, int size, char *file[]){
  int rv, k;
  int bin[10006], binM[10006];
  char FilePir[80], FileWPir[80], FileMPir[80];

  strcpy(FilePir,"./");
  strcat(FilePir,file[1]);
  FILE *fM;

  //controllo se sono nel caso singlehop o multihop
```

```
if(n > 3){
  strcpy(FileMPir,"./");
  strcat(FileMPir,file[2]);
  fM = fopen(FileMPir, "r");
}
FILE *f;
f = fopen(FilePir, "r");


strcpy(FileWPir,"./Pir.txt");
FILE *fw;
fw = fopen(FileWPir, "w");
for(k = 0; k < 10006; k++){
  bin[k] = 0;
  binM[k] = 0;
}
while(fscanf(f, "%d", &rv) == 1){
  if(rv >= 10000)
    bin[10000]++;
  else
    bin[rv]++;
}
//se sono nel caso multihop controllo anche il secondo file
if(n > 3)
  while(fscanf(fM, "%d", &rv) == 1){
    if(rv >= 10000)
      binM[10000]++;
    else
      binM[rv]++;
  }


int i, j, countS, countM, mediaS = 0, mediaM = 0, totS = 0, totM
    = 0;
for(i = 0; i < 10000; i++){
  mediaS += bin[i] * i;
  mediaM += binM[i] * i;
  totS += bin[i];
  totM += binM[i];
}
printf("Media Singlehop: %d\n", mediaS/totS);
if(n > 3)
  printf("Media Multihop: %d\n", mediaM/totM);

for(i = 0; i < 10000/size; i++){
```

```
    countS = 0;
    countM = 0;
    for(j = (i*size)+6; j <= (i+1)*size+5; j++){
      countS+=bin[j];
      countM+=binM[j];
    }

    //stampo nel file
    fprintf(fw, "%d %d %d\n", (i+1)*size, countS, countM);
  }

  fclose(f);
  if(n > 3)
    fclose(fM);
  fclose(fw);
}
```

Listing 4.9: Costruzione bin relativi ai valori di PIR ottenuti

L'esecuzione del programma prevede quindi due fasi: nella prima si compiono le operazioni sopra descritte, mostrando per ogni sotto-traccia alcune informazioni, come ad esempio la lunghezza e la quantità di pacchetti inviati e ricevuti in relazione della distanza dei veicoli fino ad un massimo di 200m. Questi primi valori calcolati ed associati alle sotto-tracce, sono molto importanti perché permettono di escludere ulteriormente dall'analisi finale situazioni particolari, come ad esempio casi in cui i veicoli si sono fermati. La seconda fase prevede la scelta dei segmenti idonei e la conseguente generazione di una serie di file di output contenenti i valori per redigere i report finali.

Capitolo 5

I risultati generali

In questo capitolo verranno descritti i risultati generali ottenuti dall'aggregazione totale dei dati. Dopo aver mostrato alcuni informazioni principali sui dati raccolti, verranno presentati tabelle e grafici relativi alle misure discusse nel capitolo precedente.

5.1 La raccolta dei dati

Le misurazioni sono state svolte durante il mese di Marzo 2012. Hanno previsto tre viaggi da Pisa a Lucca passando da Firenze (figura 5.1), ciascun viaggio lungo circa 150 Km. In totale sono stati percorsi circa 450 km, principalmente sulla superstrada FI.PI.LI. e sull'autostrada A11. La fase di post-proccesing ha operato una sostanziale scrematura di dati incorretti, lasciando circa 230 km per le operazioni di analisi.

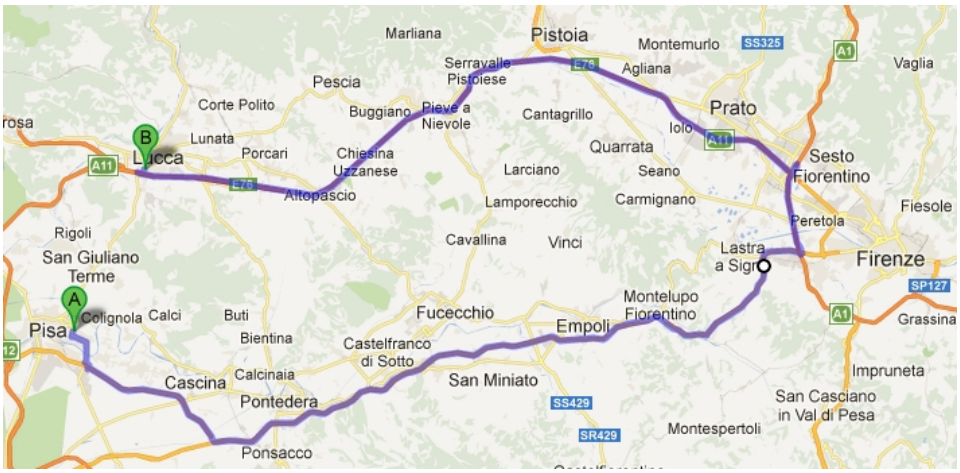

Figura 5.1: Mappa del tragitto Pisa, Firenze, Lucca (150 Km)

Ogni auto ha trasmesso nell'arco dei tre giorni circa 145000 pacchetti, circa 435000 pacchetti spediti in totale, per una dimensione totale del file di Log di circa 22 MB. In ricezione la quantità di dati ricevuta ha variato per ogni veicolo, a seconda delle condizioni stradali che ha dovuto affrontare. I valori risultanti hanno comunque mostrato un numero di pacchetti ricevuti variante tra un minimo di 115000 ad un massimo di 190000 pacchetti, circa 500000 pacchetti ricevuti in totale, con dimensioni dei file di log comprese tra i 19 MB e i 30 MB.

Ai fini dell'espirimento, è importante sottolineare che le auto in ogni viaggio sono rimaste sempre nella configurazione inziale senza superarsi. Per semplicità utilizzeremo degli ID (0,1,2), i soliti utilizzati nell'applicazione, per identificare i veicoli nell'ordine del senso di marcia.

5.2 I tipi di report

Tutte le misure verranno presentate di seguito con l'ausilio di tabelle e grafici. Vediamo in dettaglio ciascuno di essi:

- tabella relativa al *PDR*: esprime la percentuale di pacchetti ricevuti per ogni link di comunicazione. Ogni link è rappresentato nella forma $x->y$, dove x è l'ID del veicolo trasmittente e y quello ricevente. Le colonne presenti mostrano la valutazione del PDR nel caso singlehop e multihop (quest'ultimo solo nei link interessanti i veicoli 0 e 2). E' infine mostrata la distanza massima a cui i due veicoli sono stati in grado di comunicare.

- grafico relativo al *PDR vs Distance*: sull'asse delle ascisse abbiamo i valori della distanza espressi in metri, mentre sulle ordinate la percentuale di ricezione. Per ottenere questo grafico sono stati raggruppati i pacchetti ricevuti per classi di distanza con intervallo di 10 metri. Il comportamento che ci si può aspettare è una diminuzione della percentuale di ricezione, all'aumentare della distanza. Per ogni coppia di veicoli, è presente solo il grafico relativo al link nel senso di marcia

- grafico relativo al *PIR ccdf*: rappresenta il complementare della funzione di ripartizione della variabile relativa al PIR (in inglese *complementary cumulative distribution function (ccdf)*). Si calcola come $P(PIR > k)$ e misura per ogni $k > 0$, la probabilità che l'intervallo di ricezione sia maggiore di k millisecondi. E' stimata sulla base dei valori di PIR mi-

surati negli esperimenti. Sull'asse delle ascisse ci sono i valori del tempo in millisecondi mentre sulle ordinate la ccdf.

- tabella relativa al *PIR average*: rappresenta per ogni link la media, espressa in millisecondi, di tutti i valori di PIR ottenuti. Sono presenti solo i link relativi al senso di marcia.

- tabella relativa al *PIR BlackOut*: mostra la probabilità che vi sia un evento di balckout, cioè assenza di informazione aggiornata relativa ad un veicolo specifico per più di 1 secondo. Anche in questo caso sono presenti solo i link relatvi al senso di marcia.

Per ogni grafico o tabella verranno riportati commenti sull'esito delle misurazioni, aggiungendo un parametro di confronto con gli esperimenti passati. Inoltre indichiremo con link LOS quelli tra i veicoli 0-1 e 1-2 (cioè i link senza un veicolo nel mezzo), mentre con link NLOS quello tra i veicoli 0-2.

5.3 L'analisi dei risultati

L'ordine con cui verrano mostrati i risultati è il solito seguito nella fase di analisi, in quanto il calcolo di alcune misure è dipendente dal risultato di altre. Partiamo quindi col riportare i valori di PDR per ogni link.

PDR in %			
Link	Singlehop	Multihop	Max T.r. (m)
0 -> 2	29,69	57,40	290,5
2 -> 0	77,58	88,68	399
0 -> 1	70,39	–	222,5
1 -> 0	86,47	–	333,3
1 -> 2	68,78	–	159
2 -> 1	85,53	–	142,29

Tabella 5.1: Tabella coi valori di PDR per ogni link

Packet Delivery Rate (PDR)

La tabella 5.1 ci fa notare come l'utilizzo della comunicazione multihop ci permetta di incrementare la quantità di pacchetti ricevuti, con valori di incremento compresi tra 10 e 30 punti percentuali. Per quanto riguarda i valori

dei link singlehop diretti, sono conformi con quanto risultato precedentemente [6]. Da notare la presenza di asimettria nel canale comunicativo: infatti nella comunicazione opposta al senso di marcia è stata misurata una migliore qualità di ricezione. L'individuazione delle possibili cause di questo comportamento asimettrico del canale, non sono state oggetto di studio in questa tesi in quanto richiederebbero ulteriori esperimenti specifici. In ogni caso, è possibile pensare che sia frutto di condizioni legate alla struttura fisica dell'auto o a situazioni stradali e ambientali particolari.

PDR vs. Distance

Proseguiamo ora prendendo in considerazione il PDR in funzione della distanza. Questo risultato è molto importante perchè possiamo individuare un *range di trasmissione* entro il quale possiamo denotare la comunicazione come affidabile. L'identificazione di un possibile limite espresso in termini di distanza calcolata in metri, ci permette di escludere dalle analisi successive porzioni di log che contribuirebbero in modo negativo al calcolo delle altre misure. Pensiamo ad esempio ad una situazione in cui due veicoli si trovano per parecchi minuti ad una certa distanza k e tra di loro vi siano altre autovetture o mezzi pesanti. E' chiaro che in questa condizione particolare, ha poco senso includere dati che sappiamo essere frutto di uno scenario stradale complesso rispetto alla situazione in cui le due auto sono situate a breve distanza.

Le figure 5.2 e 5.3, relative ai link LOS, mostrano come all'aumentare della distanza ci sia una diminuzione di ricezione, ma sempre comunque entro buoni valori. Considerando che situazioni critiche sono al di sotto del 20% del PDR, possiamo quindi escludere l'introduzione di un limite sulla distanza. Fare un raffronto con i precedenti risultati è un'operazione poco significativa, in quanto la situazione stradale in cui si trovavano i veicoli, era sicuramente differente. La presenza di un range di trasmissione posto a 160m, non deve quindi indurre a pensare a risultati discordi, per i motivi sopra esposti.

La figura 5.4 ci mostra invece quanto sia importante l'apporto della comunicazione multihop: seppur non elevato si attesta su valori intorno al 10%, all'incirca un pacchetto in più ogni dieci. Rispetto alle due precedenti figure si nota come la qualità comunicativa cominci a peggiorare già a distanze più ravvicinate, per poi essere inadeguata oltre i 150m, sia nel caso singlehop che multihop. Questo è chiaramente una conseguenza delle diverse condizioni del canale radio fra 0 e 2, ed in particolare all'assenza di LOS (condizione NLOS).

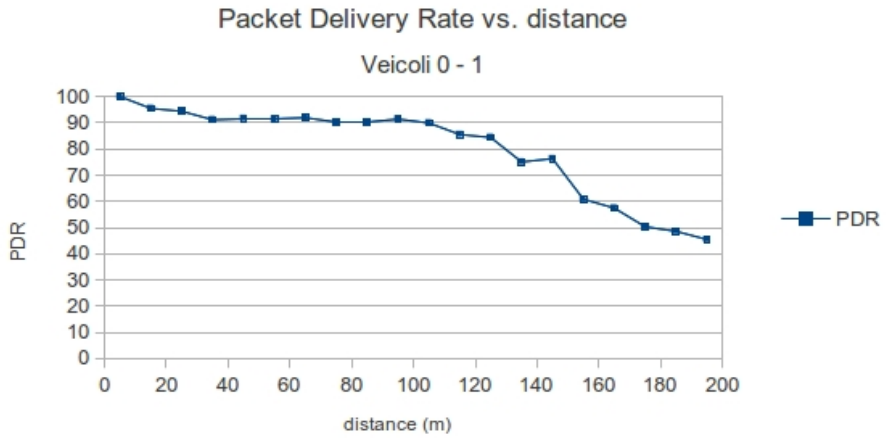

Figura 5.2: PDR in funzione della distanza tra i veicoli 0 e 1

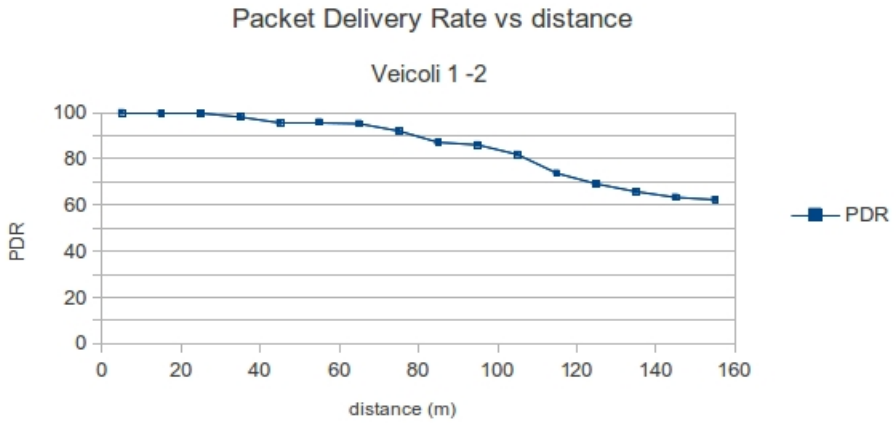

Figura 5.3: PDR in funzione della distanza tra i veicoli 1 e 2

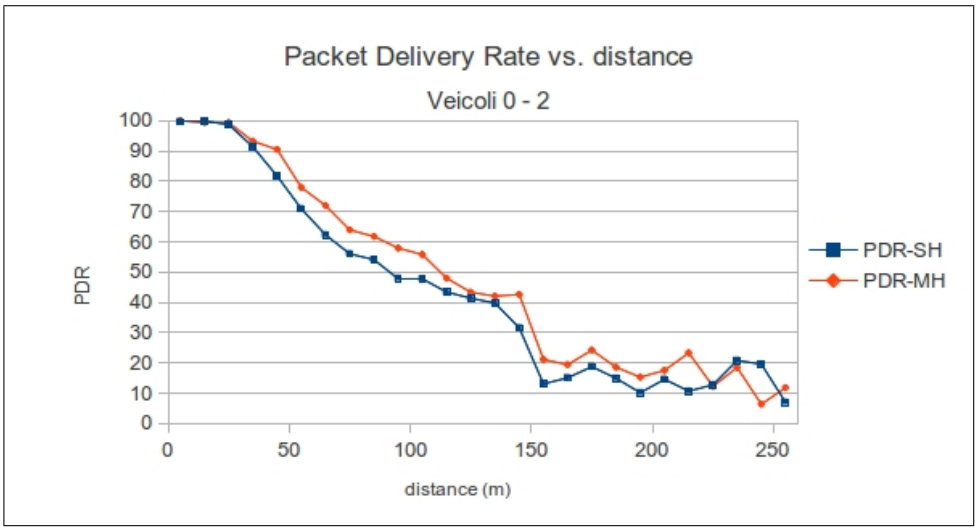

Figura 5.4: PDR in funzione della distanza tra i veicoli 0 e 2

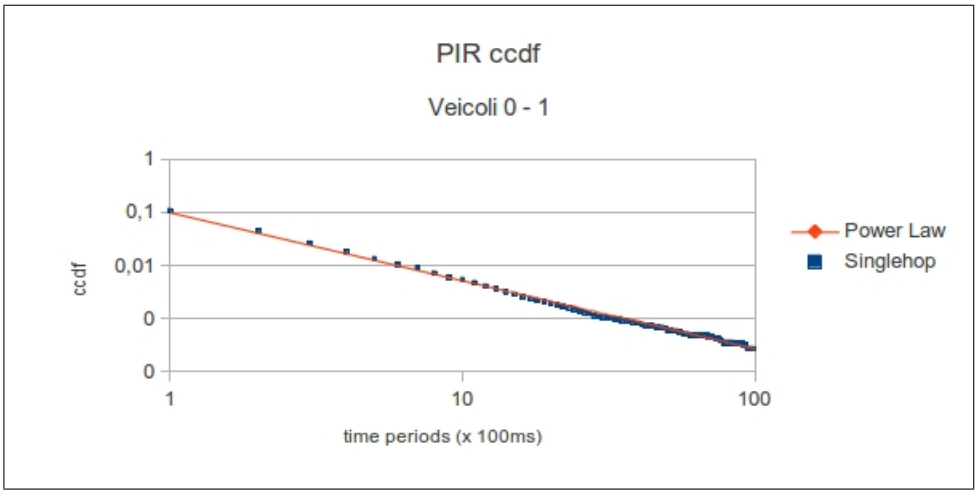

Figura 5.5: PIR ccdf tra i veicoli 0 e 1 (gli assi sono in scala logaritmica)

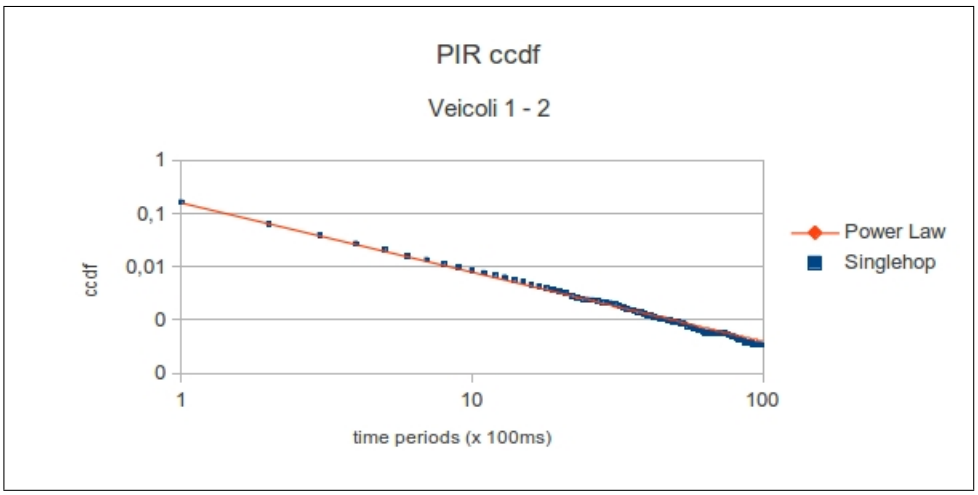

Figura 5.6: PIR ccdf tra i veicoli 1 e 2 (gli assi sono in scala logaritmica)

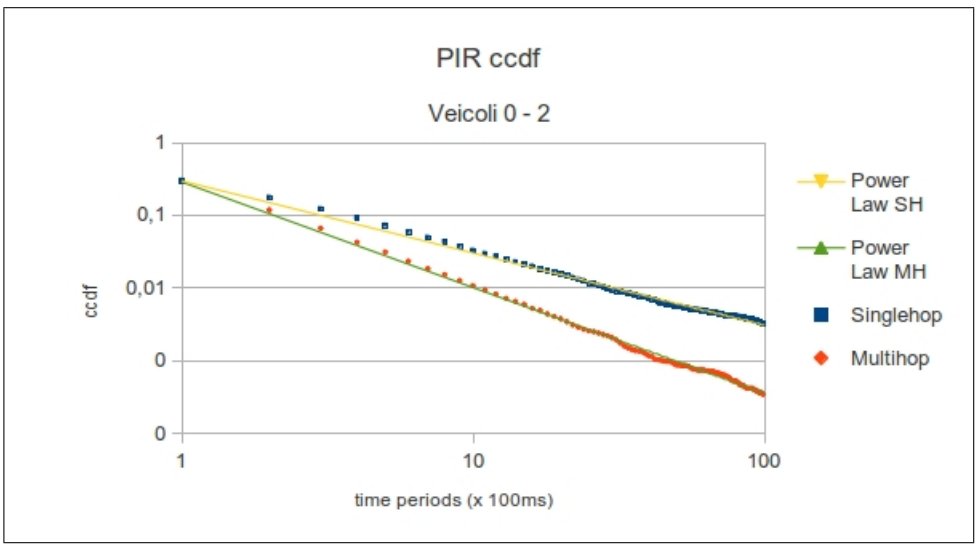

Figura 5.7: PIR ccdf tra i veicoli 0 e 2 (gli assi sono in scala logaritmica)

PIR time

Vediamo ora i risultati ottenuti nel calcolo del PIR, tenendo conto che per i link LOS non è stato definito un range di trasmissione massimo, sulla base di quanto emerso dai grafici 5.2 e 5.3, permettendo a tutti i dati di contribuire a quest'analisi. Le distribuzioni del tempo di PIR nei link LOS, sono riportate nelle figure 5.5 e 5.7. Le due distribuzioni sono altamente concentrate nel primo termine: infatti l'evento $(PIR = 1)$ ha una frequenza di 0.8928 per i veicoli 0 e 1, e 0.8583 per i veicoli 1 e 2, confermando quanto già misurato in precedenza (0.9277) [6]. I rimanenti valori di probabilità sono ben distribuiti sui i termini più grandi: questo comportamento, lineare nella scala log-log, permette di rappresentare il PIR ccdf anche attraverso una *power law*. Infatti, le seguenti power laws si adattano perfettamente ai nostri esperimenti:

$$P(PIR > k) = 0.1 \cdot (\tfrac{1}{k})^{1.28} \text{ per i veicoli 0 e 1,}$$

e

$$P(PIR > k) = 0.16 \cdot (\tfrac{1}{k})^{1.3} \text{ per i veicoli 1 e 2.}$$

Nel calcolare i tempi di PIR nel link NLOS è stato importante specificare diversamente il concetto di range massimo di trasmissione. Nel caso singlehop, è stato adottato quello definito sopra eliminando quindi tutti i dati ricevuti oltre i 150m (valore risultante dal grafico 5.4). Nel caso multihop invece, definendo con Tr_{0-1} e Tr_{1-2} il range massimo di trasmissione rispettivamente tra i veicoli 0-1 e 1-2, e con $d(0,1,t)$, $d(1,2,t)$ la distanza tra i veicoli ad ogni istante di tempo t, abbiamo concluso che:

$$\forall t, Tr_{0-2,t} = d(0,1,t) + d(1,2,t) \text{ sse } d(0,1,t) \leq Tr_{0-1} \text{ e } d(1,2,t) \leq Tr_{1-2}$$

Poiché sia Tr_{0-1} che Tr_{1-2} non sono stati definiti, non è stato necessario eliminare alcun dato per il calcolo globale del PIR multihop. La figura 5.7 denota una diminuzione della concentrazione della distribuzione sul primo termine: l'evento $(PIR = 1)$ ha infatti una frequenza di 0.69867 nel caso singlehop, e 0.70358 nel caso multihop. In questo grafico è visibile l'impatto della comunicazione multihop: infatti la curva relativa al multihop, ha lo stesso comportamento di quella delle curve nei grafici dei link LOS. Invece la curva singlehop risente della condizione comunicativa NLOS, distribuendo il resto delle probabilità principalmente non sui termini più grandi. Queste considerazione sono maggiormente rafforzate dai risultati che seguiranno, che mostreranno la media dei PIR e la probabilità di blackout.

Mostriamo infine le power laws che si adattano al caso singlehop e multihop:

$$P(PIR > k) = 0.3 \cdot (\tfrac{1}{k})^{0.99} \text{ per il caso singlehop,}$$

e

$$P(PIR > k) = 0.29 \cdot (\tfrac{1}{k})^{1.45} \text{ per il caso multihop.}$$

PIR Average e PIR Blackout

Concludiamo la nostra analisi aggregata, con due importanti metriche: il *PIR Average* e il *PIR Blackout*, mostrate rispettivamente nelle tabelle 5.2 e 5.3. In questo caso si tratta di procedere con il confronto dei valori ottenuti con quelli ottenuti nei risultati precedenti [6]. In particolare era emerso un valore di 126.29 ms per la media e 0.006 per il blackout. Osservando le tabelle, vediamo come i link LOS si mantengono sugli stessi valori, andando a confermare quanto misurato in precedenza. Per quanto riguarda il link NLOS, anche in questi risultati è possibile notare come sia fondamentale il beneficio apportato dalla comunicazione multihop, permettendo di avvicinare i valori dei link LOS. Infatti considerando un blackout come una bernoulliana con probabilità di successo 0.033 nel caso singlehop e 0.011 nel caso multihop, e assumendo l'indipendenza dei blackouts, abbiamo che i blackouts occorrono in media ogni 1/0.033 e 1/0.011 campioni, cioè ogni 4 e 12.5 secondi.

PIR Average (ms)		
Link	Singlehop	Multihop
0 -> 2	234	163
0 -> 1	129	–
1 -> 2	143	–

Tabella 5.2: Tabella coi valori medi di PIR per ogni link espressi in millisecondi

PIR Blackout		
Link	Singlehop	Multihop
0 -> 2	0.033	0.011
0 -> 1	0.005	–
1 -> 2	0.008	–

Tabella 5.3: Tabella con le probabilità di blackout (≥ 10)

5.4 Conclusioni

Secondo i risultati esposti possiamo concludere affermando quanto sia importante, nelle applicazioni di sicurezza attiva basate su beaconing, la possibilità di propagare le informazioni tra più veicoli in modalità multihop. Attraverso questo processo infatti, si riducono sensibilmente situazioni di pericolo dovute ad assenza di informazioni aggiornate su altri veicoli, posti in NLOS, per più di qualche secondo. Inoltre lo studio ci ha permesso di dare una caratterizzazione al modello L/N, evidenziando un degrado della qualità comunicativa nel caso di veicoli con assenza di LOS (condizione NLOS).

Capitolo 6

Le analisi nel dettaglio

In questo capitolo verranno presentati nel dettaglio i dati ottenuti, mostrandoli in funzione dei modelli discussi nei capitoli precedenti. Nella prima sezione analizzeremo i valori misurati nei tre singoli viaggi: saranno confrontati i risultati relativi al canale dei veicoli 0-2, ponendo l'attenzione sulle differenze tra la comunicazione singlehop e multihop in funzione del veicolo alto. Nella sezione successiva invece i dati saranno presentati a coppie di veicoli in funzione della loro altezza e delle condizioni L/N.

6.1 Confronto tra i report dei tre viaggi

I dati raccolti in ciascun viaggio, oltre ad essere stati aggregati per le analisi del capitolo precedente, sono stati post-processati anche singolarmente. Questo perché ci permette di dare una caratterizzazione più specifica in base alla configurazione delle auto utilizzate. L'analisi procederà quindi mostrando per ogni viaggio i tipi di veicoli utilizzati e il loro ordine di marcia, puntando l'attenzione sulla posizione del veicolo alto. Sulla base di questa considerazione, verranno poi rappresentati grafici e tabelle sulle stesse metriche calcolate nel capitolo precedente, mettendo in evidenza come i risultati possano variare a seconda della configurazione delle tre auto.

Nelle figure 6.1, 6.2 e 6.3 sono mostrate le configurazione delle auto in questo viaggio. In ogni report le auto sono identificathe da un numero (da 0 a 2): è possibile notare inoltre come nelle tre misurazioni l'auto alta è stata spostata nella configurazione iniziale, assumendo tutte e tre le posizioni possibili. Questo appunto per capire il comportamento della comunicazione a seconda della posizione di essa.

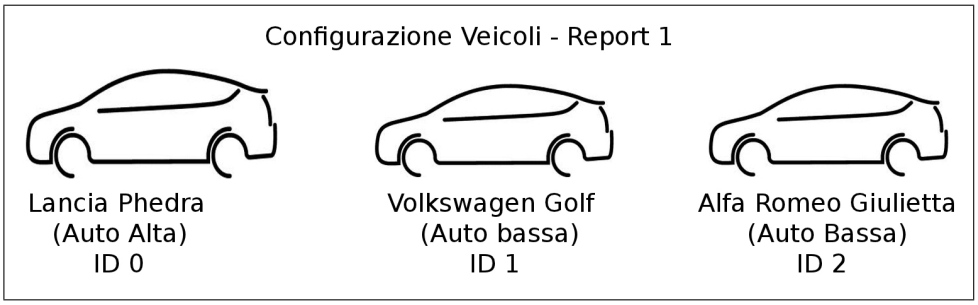

Figura 6.1: Configurazione veicoli nel primo report

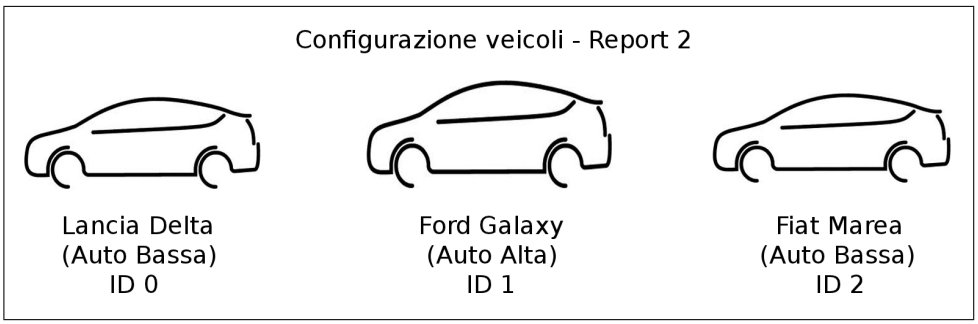

Figura 6.2: Configurazione veicoli nel secondo report

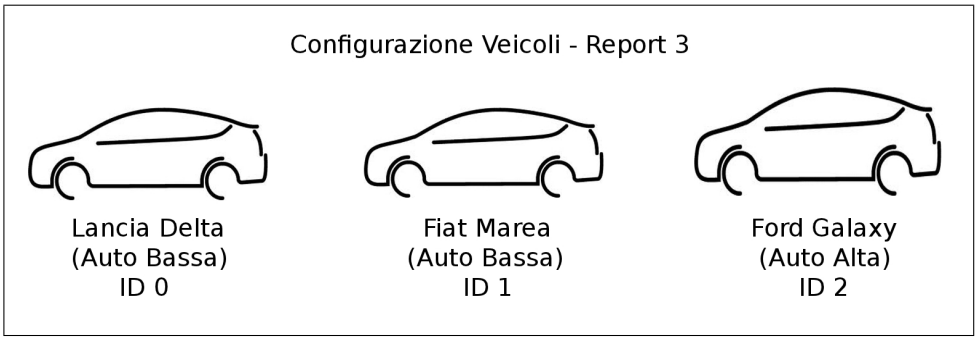

Figura 6.3: Configurazione veicoli nel terzo report

PDR

Cominciamo col confrontare i valori relativi al PDR mostrati nelle tabelle 6.1, 6.2 e 6.3. L'analisi ci mostra come in tutti e tre i casi vi siano situazioni di asimmetria del canale: nella direzione opposta al senso di marcia infatti la qualità comunicativa è stata migliore. Per quanto riguarda l'influenza del veicolo alto possiamo osservare innanzi tutto che rispetto ai veicoli bassi, quando è stato posizionato nel mezzo si è mostrato un ostacolo maggiore facendo peggiorare il PDR fino al 10%. Quando invece è stato posizionato in testa e in coda la qualità comunicativa è migliorata evidenziando però una particolarità, confermata poi nei risultati mostrati nella sezione successiva: il canale con il veicolo alto in ricezione dimostra migliori qualità rispetto all'opposto. Un'ultima analisi la facciamo sulla distanza massima a cui due veicoli hanno comunicato: in tutti e tre report i valori si attestano intorno ai 200-300 metri, valore ben distante dalle distanze massime di comunicazione nominali garantite dai laboratori NEC, sviluppatori delle *LinkBird-MX*.

PDR in %			
Link	Singlehop	Multihop	Max T.r. (m)
0 -> 2	25,99	72,84	177
2 -> 0	90,42	96,61	285,4

Tabella 6.1: Tabella coi valori di PDR per il link 0-2 del primo viaggio

PDR in %			
Link	Singlehop	Multihop	Max T.r. (m)
0 -> 2	10,52	42,72	204,2
2 -> 0	81,69	94,87	237

Tabella 6.2: Tabella coi valori di PDR per il link 0-2 del secondo viaggio

PDR in %			
Link	Singlehop	Multihop	Max T.r. (m)
0 -> 2	54,71	64,52	290,5
2 -> 0	66,72	78,28	399

Tabella 6.3: Tabella coi valori di PDR per il link 0-2 del terzo viaggio

PDR vs Distance

I grafici del PDR (figure 6.4, 6.5 e 6.6) in funzione della distanza confermano quanto evidenziato nei commenti sul PDR nel paragrafo precedente. Nei report 1 e 2, si nota come l'introduzione del multihop incrementi la qualità comunicativa con valori oltre il 30%. Nel report 3 invece la differenza tra comunicazione singlehop e multihop non è significativa, frutto probabilmente della presenza del veicolo alto come mezzo posto in ricezione nel canale. Negli altri due report invece si trova in un caso al centro, operando da ostacolo, e nell'altro in testa quindi come mezzo in invio nella configurazione del canale.

Anche in questo caso sono state fatte, per le analisi successive, tutte le considerazioni per individuare un *range di trasmissione* entro il quale possiamo denotare la comunicazione come affidabile. In tutti e tre i casi la soglia è stata posta a 150m per la comunicazione multihop, mentre nell'ordine a 70m, 50m, 150m per la comunicazione singlehop.

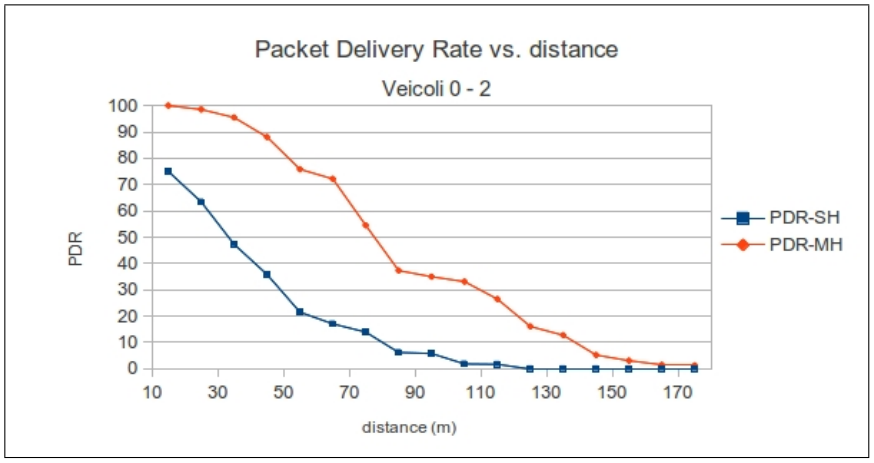

Figura 6.4: PDR in funzione della distanza tra i veicoli 0 e 2

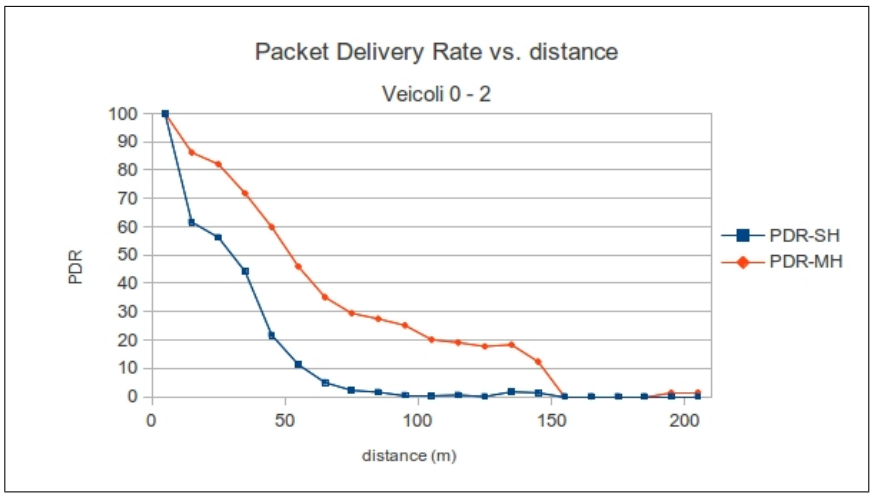

Figura 6.5: PDR in funzione della distanza tra i veicoli 0 e 2

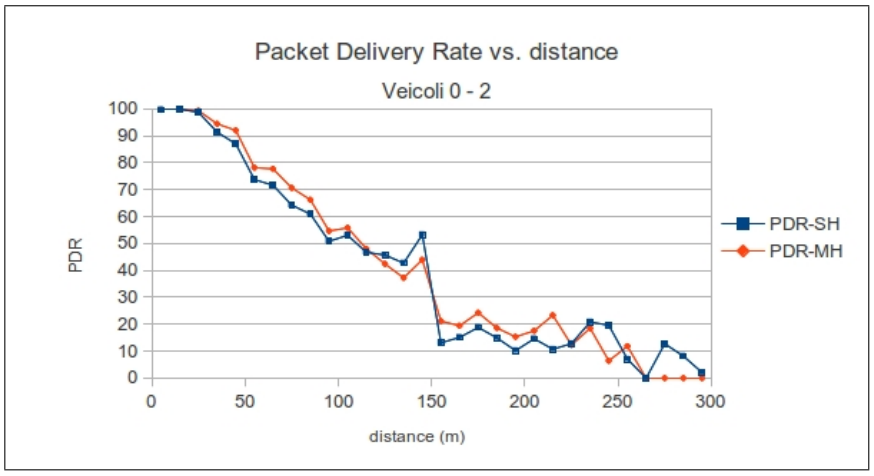

Figura 6.6: PDR in funzione della distanza tra i veicoli 0 e 2

PIR

Vediamo ora i risultati ottenuti nel calcolo del PIR, mostrati nelle figure 6.7, 6.8, e 6.9. Analizziamo prima di tutto i valori relativi all'evento ($PIR = 1$), nel singlehop e nel multihop: la frequenza è 0.57372 e 0.7389 nel report 1, 0.60267 e 0.61332 nel report 2, 0.74788 e 0.75993 nel report 3. Rispetto a quanto evidenziato nei link LOS nel capitolo precedente, abbiamo una diminuzione della concentrazione della distribuzione sul primo termine. Però ancora una volta grazie alla comunicazione multihop, la curva ha lo stesso comportamento dei link LOS, riuscendo a distribuire il resto delle probabilità sui termini più grandi a differenza della curva singlehop.

Anche in questo caso, dato il comportamento lineare nella scala log-log, abbiamo cercato di darne unna rappresentazione dei PIR ccdf attraverso delle *power laws*. E' però possibile vedere che il fitting è meno soddisfacente sopratutto nel Report 3 (figura6.9).

Per il report 1:

$$P(PIR > k) = 0.42 \cdot (\tfrac{1}{k})^{0.9} \text{ per il caso singlehop,}$$

e

$$P(PIR > k) = 0.26 \cdot (\tfrac{1}{k})^{1.6} \text{ per il caso multihop.}$$

Per il report 2:

$$P(PIR > k) = 0.39 \cdot (\tfrac{1}{k})^{0.6} \text{ per il caso singlehop,}$$

e

$$P(PIR > k) = 0.38 \cdot (\tfrac{1}{k})^{1.25} \text{ per il caso multihop.}$$

Per il report 3:

$$P(PIR > k) = 0.25 \cdot (\tfrac{1}{k})^{1.25} \text{ per il caso singlehop,}$$

e

$$P(PIR > k) = 0.24 \cdot (\tfrac{1}{k})^{1.75} \text{ per il caso multihop.}$$

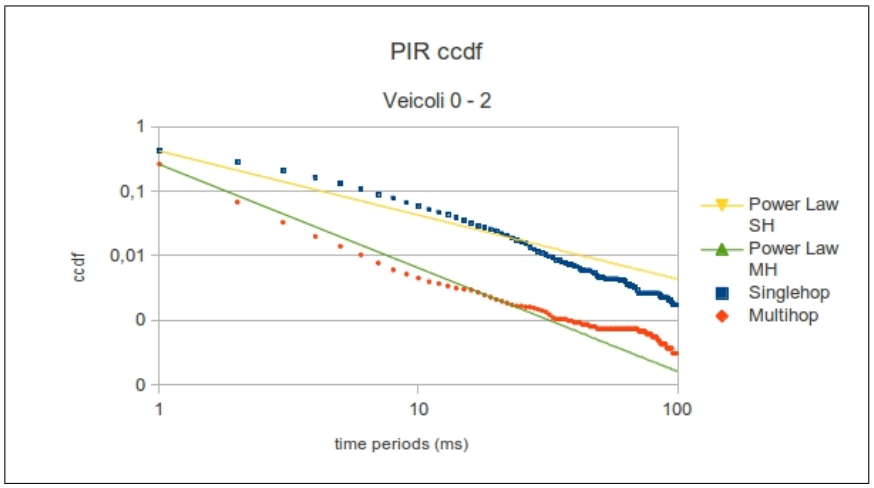

Figura 6.7: PIR ccdf nel report 1 (gli assi sono in scala logaritmica)

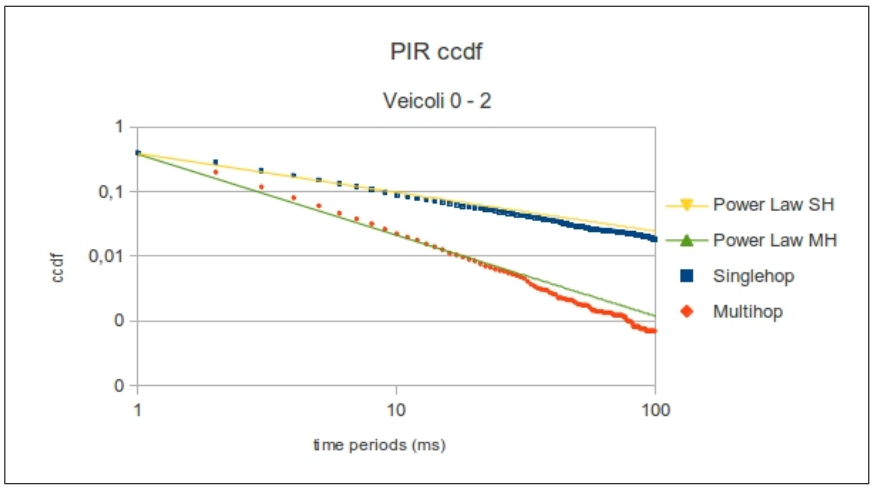

Figura 6.8: PIR ccdf nel report 2 (gli assi sono in scala logaritmica)

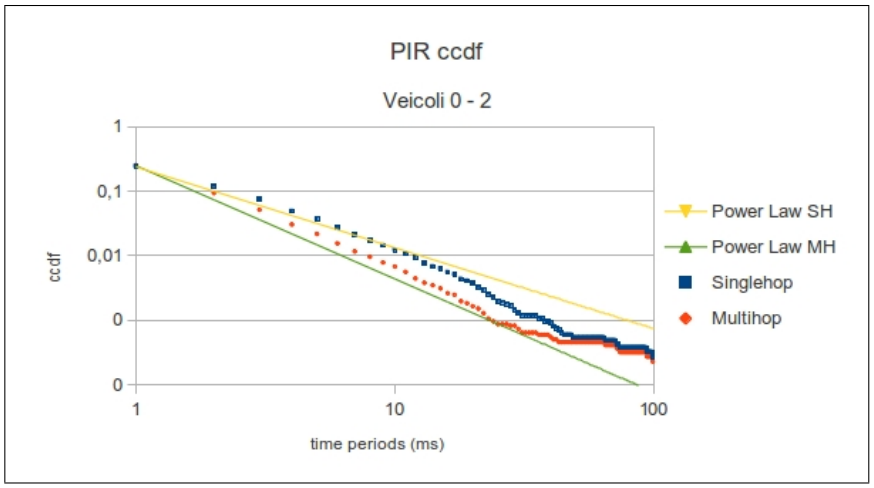

Figura 6.9: PIR ccdf nel report 3 (gli assi sono in scala logaritmica)

PIR Average e PIR Blackout

Concludiamo la nostra analisi sulle singole tracce, con le metriche del PIR Average e PIR Blackout mostrate nelle tabelle 6.4 e 6.5. Trattandosi di un link NLOS, è possibile notare come sia fondamentale il beneficio apportato dalla comunicazione multihop, permettendo di avvicinare i valori dei link LOS, mostrati nel capitolo precedente.

PIR Average (ms)		
Report	Singlehop	Multihop
1	323	133
2	399	216
3	170	144

Tabella 6.4: Tabella coi valori medi di PIR per il link 0-2 nei tre report

PIR Blackout		
Report	Singlehop	Multihop
1	0.059	0.004
2	0.087	0.022
3	0.012	0.007

Tabella 6.5: Tabella con le probabilità di blackout per il link 0-2 nei tre report

6.2 Confronto L/N sull'altezza dei veicoli

Nei dati che verranno presentati in questa sezione, sono stati considerati tutti i link diretti tra diverse coppie di veicoli in relazione alla loro altezza. Per ciascun link riporteremo l'analisi secondo il modello L/N mostrando come effettivamente varia la qualità della comunicazione in funzione della visibilità e dell'altezza. Data la presenza di un solo veicolo alto, avremmo tre tipi di link diretti: alto-basso, basso-alto e basso-basso, ciascuno considerato nel senso di marcia. I valori sono ottenuti mediante l'aggregazione dei dati, per tipo di link, misurati nei tre report.

PDR

La tabella 6.6 mostra come il canale basso-alto sia il migliore rispetto agli altri due, confermando quanto sottolineato nelle analisi nei paragrafi precedenti. Questo risultato apre le porte sicuramente a studi futuri, perché se da un lato è possibile aspettarsi migliori performance data la posizione più alta dell'antenna, dall'altro invece risulta difficile con i dati raccolti, comprendere il perché nel link opposto (alto-basso) si verifichi una così sostanziale asimettria. Per quanto riguarda invece le differenze in condizione L/N, possiamo vedere un miglioramente di circa il 30%; inoltre nei casi in cui il veicolo basso si trova in ricezione, vediamo come in LOS la percentuale di ricezione sia intorno al 55%, mentre il NLOS varia a seconda dell'altezza veicolo mittente ottenendo migliori risultati quando quest'ultimo è il veicolo alto.

PDR in %		
Link	LOS	NLOS
Alto -> Basso	55,64	25,99
Basso -> Alto	88,43	54,71
Basso -> Basso	55,08	10,52

Tabella 6.6: Tabella con i valori di PDR nelle coppie di veicoli distinti per altezza

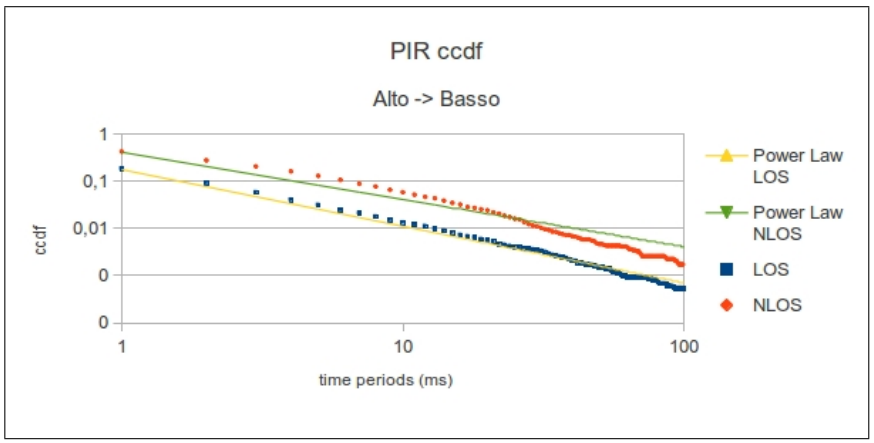

Figura 6.10: PIR ccdf nel link alto-basso (gli assi sono in scala logaritmica)

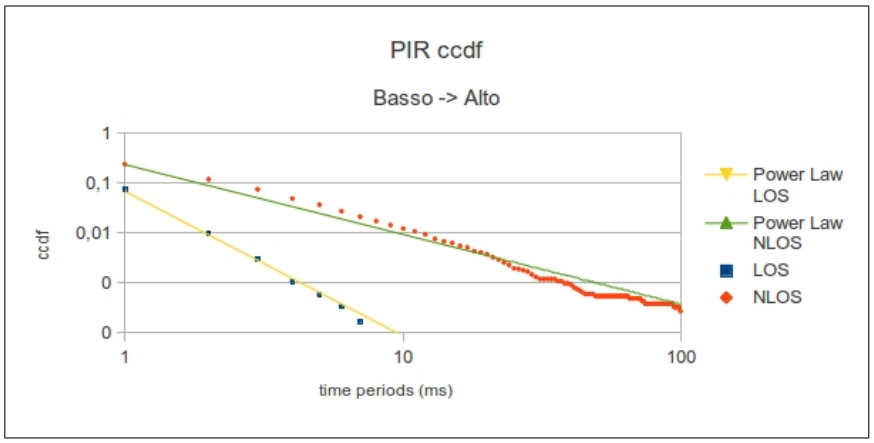

Figura 6.11: PIR ccdf nel link basso-alto (gli assi sono in scala logaritmica)

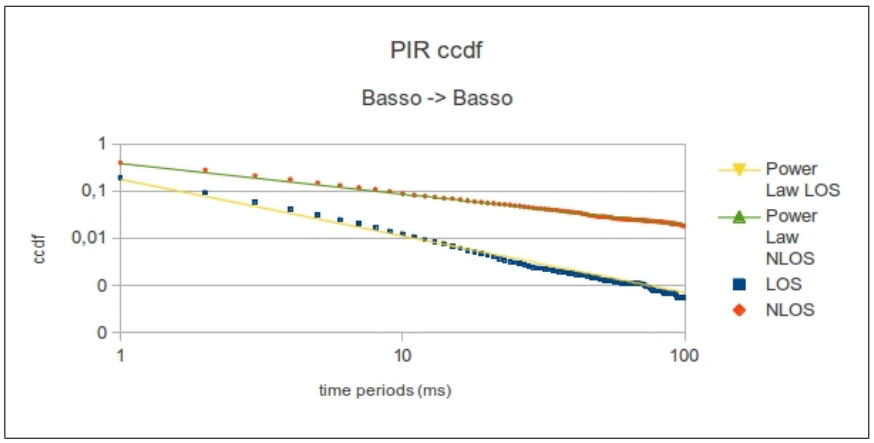

Figura 6.12: PIR ccdf nel link basso-basso (gli assi sono in scala logaritmica)

PIR

Vediamo ora i risultati del PIR mostrati nelle figure 6.10, 6.11 e 6.12. Questi sono in linea con le percentuali di PDR mostrate nelle tabelle precedenti. Infatti se consideriamo l'evento $(PIR = 1)$ otteniamo i valori per LOS e NLOS rispetttivamente di 0.81527 e 0.57352 nel link alto-basso, 0.92645 e 0.75993 nel link basso-alto, 0.81126 e 0.60267 nel link basso-basso. Anche in questo caso osserviamo come la distribuzioni delle probabilità nel caso LOS sia principalmente concentrata nel primo termine e il resto nei termini successivi. Addirittura nel caso del link basso-alto la curva assume un comportamento diverso, subendo una rapida decrescita già sui primi termini, conformandosi con tutte le altre considerazioni su questo tipo di link. Nel caso NLOS invece la distribuzione delle probabilità è simile a quella osservata nel link singlehop 0-2 nel capitolo precedente, quindi non quasi totalmente distribuita sui primi termini.

Mostriamo infine le power laws le quali riescono avere un miglior fitting rispetto alla sezione precedente, anche se alcune, specialmente per il caso NLOS, non sono proprio perfette.

Per il link alto-basso:

$$P(PIR > k) = 0.18 \cdot (\tfrac{1}{k})^{1.2} \text{ per il caso LOS,}$$

e

$$P(PIR > k) = 0.42 \cdot (\tfrac{1}{k})^{1} \text{ per il caso NLOS.}$$

Per il link basso-alto:

$$P(PIR > k) = 0.07 \cdot (\tfrac{1}{k})^{2.9} \text{ per il caso LOS,}$$

e

$$P(PIR > k) = 0.24 \cdot (\tfrac{1}{k})^{1.4} \text{ per il caso NLOS.}$$

Per il link basso-basso:

$$P(PIR > k) = 0.18 \cdot (\tfrac{1}{k})^{1.2} \text{ per il caso LOS,}$$

e

$$P(PIR > k) = 0.39 \cdot (\tfrac{1}{k})^{0.65} \text{ per il caso NLOS.}$$

PIR Average e PIR Blackout

Chiudiamo anche in questo caso con le tabelle di PIR Average (6.7) e PIR Blackout (6.8). I valori sono coerenti sia con i risultati degli esperimenti precedenti [6], sia con quanto descritto nei paragrafi sopra. Da sottolineare ancora una volta quanto il link basso-alto si avvicini all'ottimo, con valori medi di PIR di 105ms e una probabilità di blackout di 0.0005, oltre quindi i 100s.

PIR Average (ms)		
Link	LOS	NLOS
Alto -> Basso	163	323
Basso -> Alto	105	170
Basso -> Basso	191	399

Tabella 6.7: Tabella con i valori medi di PIR nelle coppie di veicoli distinti per altezza

PIR Blackout		
Link	LOS	NLOS
Alto -> Basso	0.01	0.06
Basso -> Alto	0.0005	0.01
Basso -> Basso	0.01	0.09

Tabella 6.8: Tabella con le probabilità di blackout nelle coppie di veicoli distinti per altezza

6.3 Conclusioni

Secondo quanto analizzato in questo capitolo possiamo affermare quanto sia difficile definire un comportamento preciso per la comunicazione veicolare. Abbiamo infatti visto che anche la differente altezza dei veicoli e le condizioni di visibilità diretta o no, incidano sulla qualità comunicativa, andando ad aggiungere altre variabili ad uno scenario che di per sé ne presenta già molte. Nel nostro caso possiamo solo dire che sono state osservate delle migliori performance per quanto riguarda il link col veicolo alto in ricezione, avvicinandosi ad una situazione di ottimo.

Capitolo 7

Conclusioni

L'obiettivo primario di questa tesi era valutare quanto potesse incidere nell'ambiente veicolare lo scambio di beacon tramite comunicazione multihop garantendo i requisiti richiesti per lo sviluppo di applicazioni nell'ambito della sicurezza attiva. In aggiunta a questo obiettivo, ne sono stati considerati degli altri, come la qualità del canale secondo il modello L/N o le problematiche annesse alla differente altezza dei veicoli. Per rispondere a questo interrogativo, con l'utilizzo delle unità *LinkBird-MX* è stato realizzato in *Java* un programma per lo scambio periodico di beacon tra più veicoli e un programma di Post-Processing per analizzare i risultati ottenuti. Grazie alle misurazioni effettuate, che hanno permesso di raccogliere una buona quantità di dati, è stato possibile rispondere alla nostre domande.

Innanzi tutto abbiamo visto che effettivamente la possibilità di avere uno scenario in cui ciascun veicolo contribuisce alla conoscenza cooperativa tramite l'invio di beacon con le informazioni di tutti i veicoli, è positivo. Infatti permette di avere un incremento del numero di pacchetti ricevuti, valore che determina l'intervallo di ricezione tra i vari pacchetti, diminuendo situazioni in cui il tempo di aggiornamento della conoscenza supera valori critici. Inoltre abbiamo valutato come la presenza di un veicolo alto permetta da un lato una migliore propagazione delle informazioni, ma dall'altro costituisce un ostacolo per eventuali comunicazioni dirette. Infine abbiamo rilevato come nel caso di canali comunicativi tra un veicolo alto ed uno basso, risulta essere più performante il caso in cui il veicolo alto si trovi in ricezione.

7.1 Lavori futuri

Sulla base di quanto sviluppato e analizzato, possibili estensioni possono riguardare ad esempio l'implementazione di una funzione di refresh della tabella dei veicoli valutando quindi quanto debba essere l'intervallo di tempo minimo dopo il quale un veicolo non più incontrato possa essere rimosso. Altri interessanti studi potrebbero riguardare le caratteristiche del canale radio: individuare quindi le cause dell'asimettria di esso oppure capire quanto incidano fattori quali la tipologia dei veicoli o l'ambiente circostante.

7.2 Esperienze acquisite

Oltre ad aver acquisito maggiore esperienza nella programmazione nei linguaggi *Java* e *C*, il lavoro svolto in questa tesi mi ha permesso di conoscere i sistemi veicolari capendone la struttura e i problemi, in particolare per quel che riguarda le applicazioni di sicurezza attiva basate su beaconing. Inoltre il lavoro effettuato mi ha fornito le basi per capire in quale maniera svolgere delle analisi sulla qualità della comunicazione in suddetti ambienti.

Bibliografia

[1] H. Hartenstein, K. Labertaux. *VANET - Vehicular Applications and Inter-Networking Tehnologies*. WILEY, 2010.

[2] C. K. Toh. *Ad-Hoc Mobile Wireless Network - Protocols and Systems*. Prentice Hall, NJ, 2002.

[3] Marco Volpetti. Applicazione VoIP su reti veicolari IEEE 802.11p: misurazioni ed analisi delle prestazioni. Master's thesis, Università di Pisa, 2011.

[4] Savas Konur and Michael Fisher. Formal Analysis of a VANET Congestion Control Protocol through Probabilistic Verification. In *Proc. 73rd IEEE Vehicular Technology Conference (VTC2011-Spring)*, May 2011.

[5] Hao Jang and Siyue Chen and Yang Yang and Zhizhong Jie and Henry Leung and Jun Xu and Lin Wang. Estimation of packet loss rate at wireless link of VANET. *6th International Conference on Wireless Communications Networking and Mobile Computing (WiCOM)*, 2010.

[6] Francesca Martelli, M. Elena Renda, Giovanni Resta, and Paolo Santi. A measurement-based study of beaconing performance in IEEE 802.11p vehicular networks. *IEEE International Conference on Computer Communications (IEEE INFOCOM 2012), Orlando, Florida (USA)*, 2012.

[7] E. O. Elliot. Estimates of error Rates for Codes on Burst-Noise Channels. *Bell Syst. Tech. J.*, 42:1977–1997, 1964.

[8] E. N. Gilbert. Capacity of Burst-Noise Channel. *Bell Syst. Tech. J.*, 39:1253–1265, 1960.

[9] S. Eichler. Perfomance Evaluation of the IEEE 802.11p WAWE Communication Standard. In *Proceedings of the 1st IEEE International Symposium on Wireless Vehicular Communications (WiVeC)*, 2007.

[10] M. Sepulcre, J. Mittag, P. Santi, H. Hartenstein, and J. Gozalvez. Congestion and Awareness Control in Cooperative Vehicular Systems. In *Proceedings of the IEEE*, volume 99, pages 1260–1279, 2011.

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

[12] V. Shivaldova and G. Maier and D. Smely and N. Czink and A. Alonso and A. Wilkenbauer and A. Paier and C. F. Mecklenbräuker. Perfomance Evaluation of IEEE 802.11p Infrastucture-to-Vehicle Tunnel Measurements. In *Proc. The 7th International wireless Communications and Mobile Computing Conference (IWCMC-2011)*, July 2011.

[13] Cristoph Sommer and David Eckhoff and 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)*, January 2011.

[14] Moez Jerbi and Patrick Marlier and Sidi Mohammed Senouci. Experimental Assessment of V2V and V2I Communications. *IEEE Internatonal Conference on Mobile Adhoc and Sensor Systems*, 2007.

[15] J.P. Singh and N. Bambos and B. Srinivasan and D. Clawin. Wireless lan performance under varied stress conditions in vehicular traffic scenarios. In *Proc. IEEE 56th Vehicular Technology Conference, Fall 2002*, 2002.

[16] Francesca Martelli, M. Elena Renda, and Paolo Santi. Measuring IEEE 802.11p Performance for Active Safety Applications in Cooperative Vehicular Systems. *IEEE 73rd Vehicular Technology Conference (VTC-Spring)*, May 2011.

[17] Miguel Sepulcre Ribes. *Adaptive Communication Protocols for Cooperative Vehicular Systems.* PhD thesis, Universidad Miguel Hernandez de Elche, 2010.

[18] T. ElBatt, S. K. Goel, G. Holland, H. Krishnan, and J. Parikh. Cooperative Collision Warning Using Dedicated Short Range Wireless Communications. *Proc. ACM VANET*, 2006.

[19] X. Yang, J. Liu, F. Zhao, and N.H. Vaidya. A Vehicle-to-Vehicle Communication Protocol fo Cooperative Collision Warning. *Proc. IEEE MobiQuitous*, 2004.

[20] F. Bai, D. D. Stancil, and H. Krishnan. Toward Understanding Characteristics of Dedicated Short Range Communications (DSRC) Form a Perspective of Vehicular Network Engineers. *Proc. ACM Mobicom*, 2011.

[21] K. Hong, D. Xing, V. Rai, and J. Kenney. Characterization of DSRC Performance as a Function of Transmit Power. *Proc. ACM VANET*, 2009.

[22] M. Torrent-Moreno, J. Mittag, P.Santi, and H. Hartenstein. Vehicle-to-vehicle communication: Fair transmit power control for safety-critical information. *IEEE Tr. on Vehicular Technology*, September 2009.

[23] Car-2-Car Communication Consortium. C2C-CC Manifesto. 2007. Versione 1.1.

[24] J. B. Kenney. *Standards and Regulations in VANET: Vehicular Applications and Inter-Networkng Technologies.* John Wiley and Sons, Chichester, UK, 2009.

[25] ECC decision of march 2008 on the harmonised use of the 5875-5932 mhz frequency band for intelligent transport system (its). 2008. `http://www.erodocdb.dk`.

[26] 802.11p Standards. `http://standars.iee.org/getieee802/download/802.11p-2010.pdf`.

[27] ETSI TC ITS. Intelligent Trasport Systems (ITS); European profile standard on the pysical and medium access layer of 5 GHz ITS. *Draft ETSI ES 202 663 V0.0.6*, October 2009.

[28] The WAVE Communications Stack: IEEE 802.11p, 1609.4 and 1609.3, Technomm, September 2007.

[29] 802.11 Standards. `http://standars.iee.org/about/get/802/802.11.html`.

[30] E. Schoch, F. Kargl, M. Weber, and T. Leinmuller. Communication patterns in VANETs. *IEEE Communications Magazine*, November 2008.

[31] ETSI TC ITS. Intelligent trasport systems (its); vehicular communications; basic set of application; definitions. *ETSI TR 102 638 V1.1*, June 2009.

[32] VSC Consortium. Vehicle Safety Communications Project Task 3 - Finale Report: Identify Intelligent Vehicle Safety Applications Enables by DSRC. *DOT HS 809 859*, March 2005.

[33] Network Division NEC Laboratories:. CAR-2-X Communication SDK. User Guide. NEC Europe Ltd., 29.10.2009 Version 1.5.2.

[34] I. A. Sumra, H. B. Hasbullah, and J. Ab Manan. Comparative study of security hardware module (EDR, TPD and TPM) in VANET. *3rd National Information Technology Symposium (NITS 2011)*, 2011.

[35] P. Papadimitratos, L. Buttyan, T. Holczer, E.Schoch, J. Freudiger, M. Raya, Z. Ma, F. Kargl, A. Kung, and J.P. Hubaux. Secure Vehicular Communication System: Design and Architecture. *IEEE Communications Magazine*, November 2008.

[36] Car-2-Car Communication Consortium official website. `http://www.car-2-car.org`.

[37] A. Festag, R. Baldessari, W. Zhang, and L. Le. CAR-2-X Communication SDK - A Software Toolkit for Rapid Application Development and Experimentations. *IEEE VehiMobil 2009*.

[38] F. Kargl, Gerlach M, and T. LeinMuller. Security and Privacy for C2X Communication System - Research and Standards. 2009. 4th ETSI Security Workshop.