

ALMA MATER STUDIORUM · UNIVERSITÀ DI
BOLOGNA

FACOLTÀ DI SCIENZE MATEMATICHE, FISICHE E NATURALI

Corso di Laurea in Scienze di Internet

**Progettazione e implementazione di
componenti CLIENT e SERVER per
la gestione di dati sulla viabilità
ottenuti tramite reti veicolari**

Tesi di Laurea in Sistemi e Reti Wireless

Relatore:
Chiar.mo Prof.
LUCIANO BONONI

Presentata da:
STEFANO SUSI

Sessione seconda
Anno Accademico 2006-2007

*Alla mia famiglia
ed agli amici di sempre
Rino Michelino e Vittorio.*

Indice

1	Introduzione	1
2	I sistemi GNSS	5
2.1	Storia del GPS	6
2.2	I segmenti del sistema GPS	8
2.2.1	Segmento spaziale	9
2.2.2	Segmento di Controllo Terrestre	10
2.2.3	Segmento Utente	11
2.3	Frequenze Utilizzate e codici trasmessi	15
2.3.1	Codice P	15
2.3.2	Codice C/A	15
2.3.3	Navigation Data D	16
2.4	Fonti di errore	18
2.4.1	Errori dovuti alla propagazione dei segnali nella Tro- posfera e nella Ionosfera	18
2.4.2	Errori dovuti a percorsi multipath	20
2.4.3	Errore della posizione nello spazio dei satelliti	21
2.4.4	Errori dovuti a ostacoli ambientali	21
2.4.5	Degradazione volontaria del segnale	22
2.5	Galileo	23

2.5.1	Servizi Offerti	26
3	Le Reti Wireless	31
3.1	Reti Wireless	31
3.1.1	Interferenza e Affidabilità	32
3.1.2	CSMA/CA	32
3.1.3	Terminale nascosto	33
3.2	Reti MANET	35
3.2.1	Alta mobilità dei nodi	35
3.2.2	Assenza del controllo e della gestione della rete.	36
3.3	Reti VANET	38
3.3.1	Protocolli di Routing	40
3.4	802.11p	48
3.4.1	PHY Layer	49
3.4.2	MAC Layer	52
3.4.3	Conclusioni	53
4	Il progetto	55
4.1	La piattaforma LAMP	56
4.1.1	Come funziona una tipica applicazione LAMP?	56
4.2	Ottimizzazione componenti LAMP	58
4.2.1	Ottimizzazione LINUX	58
4.2.2	Ottimizzazione Apache	63
4.2.3	Ottimizzazione PHP	67
4.2.4	Ottimizzazione MySQL	70
4.3	Componente CLIENT	73
4.4	Componente SERVER	79
4.4.1	Upload dei file KML	79

<i>INDICE</i>	5
4.4.2 ViaMichelin Web Services	79
4.4.3 DataBase	83
4.4.4 I servizi offerti	84
4.5 Sviluppi futuri	86
5 Conclusioni	87
A Proto-Codice Applicazione	89
A.1 Client	89
A.1.1 Load KML	89
A.1.2 Crea KML	93
A.2 Server	98
A.2.1 Reverse Geocoding	98
A.2.2 Info Viabilità	103

Elenco delle figure

2.1	I satelliti del Sistema GPS	8
2.2	I segmenti del Sistema GPS	9
2.3	La posizione delle stazioni terrestri	10
2.4	La struttura del Navigation Data Message	17
2.5	Errore dovuto alla variazione di velocità nella Ionosfera	19
2.6	Il segnale ricevuto dai satelliti a bassa elevazione	19
2.7	Errori causati dal multipath	20
2.8	Buona e pessima geometria dei satelliti	21
2.9	I componenti del Sistema GALILEO	28
3.1	Terminale nascosto RTS/CTS	33
3.2	Ad Hoc NetWorks	38
3.3	Formula del Defertime	46
3.4	Gli stati dei messaggi di allarme	47
3.5	Attivazione dei messaggi di allarme	48
3.6	Modulazione FDM/OFDM	49
3.7	802.11p Livello Fisico	50
4.1	Tipica architettura LAMP	57
4.2	Esempio di attacco SYN flood	62
4.3	Tipologia di Web Server Utilizzati	64

4.4	Confronto php cache	68
4.5	Visualizzazione del KML iniziale	74
4.6	Calcolo della distanza	76
4.7	Visualizzazione del KML finale	77
4.8	ViaMichelin Web Services	80
4.9	SAJAX Toolkit	82
4.10	Richiesta info stato viabilità	84
4.11	Risposta del server su stato viabilità	85

Capitolo 1

Introduzione

Oggetto della tesi è la realizzazione di un'applicazione che permetta di sfruttare due tecnologie già esistenti sul mercato: il sistema di posizionamento GPS e le reti wireless.

Negli ultimi anni vi è stato un aumento del traffico nelle autostrade, strade urbane ed extraurbane che ha comportato un aumento della difficoltà nella guida per i conducenti di veicoli di qualsiasi tipo.

Tali disagi comportano un maggiore stress e quindi crescono i rischi ad esso connessi (incidenti, ostacoli, imprevisti). La ricerca scientifica effettuata da enti statali, aziende private, multinazionali sta dando numerosi frutti in aiuto alle nuove problematiche di sicurezza.

In prima istanza sono stati realizzati dispositivi basati sul veicolo fisico. Si pensi ai sistemi di assorbimento urti come airbag o barre laterali. Inoltre sono stati sviluppati sistemi che migliorano la stabilità e l'assetto della vettura (ESP) e sistemi di miglioramento della frenata (ABS). Fino ad ora scarsa importanza era stata data alla cooperazione tra i veicoli.

Grazie all'ausilio dei sistemi di posizionamento e reti wireless sempre più efficienti, attualmente è possibile studiare gli aspetti di cooperazione tra i

nodi.

I veicoli che percorrono una data strada, inviano ai propri vicini (veicoli che si trovano all'interno dell'area di copertura del raggio d'azione del trasmettente) informazioni di vario tipo come ad esempio messaggi di allerta. Inoltre hanno la possibilità di comunicare con un server per fornire dati riguardanti la propria velocità e direzione, il tipo di strada, il traffico presente, etc.

Particolare attenzione è stata prestata alla comunicazione di messaggi di allarme tra i vari nodi: sono stati analizzati diversi protocolli che permettono di propagare il messaggio attraverso i nodi della rete, che essendo una particolare rete MANET non presenta un'infrastruttura fissa.

La ricerca è indirizzata alla realizzazione di componenti che possano informare il conducente sullo stato di viabilità. È necessario poter inviare ad un server le rilevazioni di posizione attraverso la rete Internet. I dispositivi dovranno essere equipaggiati sia con un ricevitore GPS sia con un sistema operativo che permetta di utilizzare un'applicazione java.

Lato server, sarà possibile ricevere le informazioni e immagazzinarle in un database. Successivamente si potranno fornire indicazioni utili sullo stato del traffico in tempo reale, nonché consigliare percorsi alternativi.

Il presente lavoro è suddiviso in sei capitoli:

Il Capitolo 2 introduce brevemente il sistema satellitare GPS, descrivendone la storia, le caratteristiche e le possibili alternative. Verranno analizzati i principi fondamentali di funzionamento per fornire una visione di insieme delle parti che compongono l'intero progetto.

Il Capitolo 3 presenta le reti VANET come sotto categoria delle reti MANET. In generale verranno affrontate problematiche relative alla comunicazione tra i vari nodi e presentate alcune possibili soluzioni. Sarà analizzato lo

standard 802.11p che permette una comunicazione più efficiente in contesti veicolari.

Il Capitolo 4 descrive il progetto di tesi. Lo studio sarà incentrato sull'architettura utilizzata (LAMP), l'ottimizzazione di quest'ultima, l'iterazione tra i vari componenti e l'analisi dei risultati ottenuti.

Nel Capitolo 5 verranno tratte delle conclusioni generali e indicati possibili scenari futuri.

In appendice è stato inserito il codice dell'applicazione sviluppata.

Capitolo 2

I sistemi GNSS

I sistemi di navigazione satellitare globale, conosciuti anche come GNSS, rispondono a particolari esigenze dell'uomo come conoscere la propria posizione sulla terra o un possibile cammino tra due punti. Attualmente vengono utilizzati il GLONASS russo e soprattutto il GPS statunitense.

Questi sistemi nascono per esigenze militari e solo successivamente sono stati rivolti ad un utilizzo civile, il GPS nel 1993 mentre il sistema russo due anni dopo. La loro origine militare comporta diversi problemi legati all'accesso ai servizi.

In questo capitolo verranno illustrati la storia ed il funzionamento del sistema di posizionamento GPS. Inoltre verrà analizzato il sistema di posizionamento Galileo sviluppato in Europa come alternativa al GPS, che viene controllato dall'esercito americano. Al contrario il sistema GALILEO nasce per un utilizzo civile e prevede di garantire un servizio affidabile, indipendente da scelte politiche o militari.

2.1 Storia del GPS

Uno dei primi sistemi di posizionamento fu il TRANSIT originariamente disegnato dalla Marina Militare degli Stati Uniti d'America(Navy), il suo scopo era quello di localizzare missili balistici, sottomarini ed altre navi presenti in mare [gps]. Il primo satellite venne lanciato nel 1959, ma non raggiunse mai l'orbita di destinazione a causa del mancato avviamento del terzo stadio del razzo vettore. Solo nel 1960 il satellite raggiunse la sua orbita polare a 1100 Km di distanza dalla Terra. Nel 1967 fu reso disponibile per usi civili. Tale sistema, denominato Navy Navigation Satellite System (NNSS) è costituito da un gruppo di sei satelliti identici posti in orbite diverse. Sebbene il sistema fosse molto utile per la navigazione marina presentava alcuni difetti:

- Risultava lento.
- Richiedeva molto tempo per effettuare la localizzazione.
- Aveva la capacità di localizzare solo due dimensioni (longitudine e latitudine).
- Il segnale presentava frequenti salti.
- Gli utenti dovevano correggere la loro posizione in funzione della velocità di navigazione.

Tutto questo rendeva praticamente impossibile l'utilizzo di TRANSIT in contesti aerei o in cui vi fosse stato un rapido movimento.

Successivamente venne concepito e sviluppato il sistema TIMATION, lanciato in orbita dal Naval Research Laboratory agli inizi del 1964. Il concetto alla base di questo sistema era quello di fornire in broadcast una stima accurata del *time reference*, per poter essere ricevuto sulla Terra.

Il secondo satellite TIMATION fu lanciato nel 1969 provvisto di un orologio atomico che permetteva di avere una precisione maggiore nella previsione dell'orbita satellitare. Nello stesso periodo l'Air Force stava lavorando ad un programma molto simile, denominato System 612B, che riusciva a fornire tre dimensioni (latitudine, longitudine ed altezza). La tecnica utilizzata, denominata Pseudo Random Noise (PRN), permetteva di individuare la posizione del mezzo con una precisione massima di un centesimo di miglio.

Per coordinare gli sforzi di tutti questi soggetti, il Dipartimento della Difesa Americano (DoD) stabilì di riunire i tre organi nel 1968 in un comitato chiamato NAVSEG (Navigation Satellite Executive Group). Questo organo spese molti degli anni successivi nello stabilire le specifiche del sistema di navigazione, come per esempio il numero di satelliti da impiegare, l'altitudine, la modulazione del segnale etc...

Proprio da questo comitato nacque il sistema di posizionamento GPS.

2.2 I segmenti del sistema GPS

Il sistema è formato da una costellazione di 24 satelliti, di cui 3 non attivi, che permette di localizzare un soggetto su 3 coordinate (latitudine, longitudine ed altitudine) [Gar].

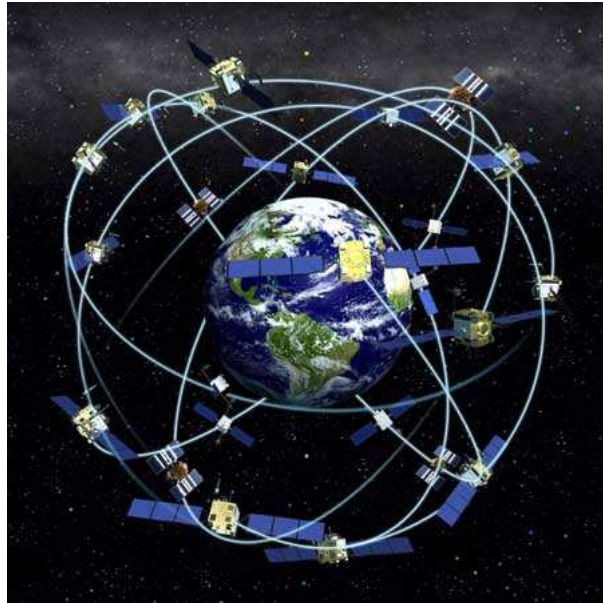


Figura 2.1: I satelliti del Sistema GPS

Può essere diviso in tre grandi componenti:

1. Il segmento spaziale
2. Il segmento operativo e di controllo
3. Il segmento utente



Figura 2.2: I segmenti del Sistema GPS

2.2.1 Segmento spaziale

I 24 satelliti formano quello che è denominato Space Segment, essi trasmettono codici di distanza e dei dati di navigazione. Distribuiti su sei diversi piani orbitali, formano un angolo di 55° rispetto al piano equatoriale ed uno di 60° tra gli altri piani orbitali. I quattro satelliti di ciascun piano sono distanziati in maniera uniforme, tuttavia questa misura può essere modificata dal centro di controllo terrestre al verificarsi di particolari condizioni.

Le loro orbite risultano circolari ed effettuano l'intero percorso in un periodo di 12 ore, la costellazione è stata disegnata per avere la presenza di almeno 4 satelliti sempre visibili in qualsiasi punto della Terra. Per riuscire nel loro intento, ovvero quello del posizionamento, ogni satellite trasmette due onde radio portanti, modulate da altrettanti codici pseudo-casuali, il C/A (coarse-acquisition) ed il codice P(precision).

Altra caratteristica molto importante dei satelliti è quella di avere orologi atomici (sia al cesio che al rubidio) che permettono di ottenere un riferimento temporale molto preciso ed accurato per i segnali radio.

La quota teorica è di 20.200 Km e permette una visione ottimale da parte

degli utilizzatori finali, quindi è sempre possibile effettuare la navigazione tridimensionale (latitudine, longitudine e quota). Solitamente il tempo di vita medio di un satellite è di circa 7 anni e, come vedremo successivamente, esiste un campo del pacchetto dati trasmesso che informa i segmenti a terra sullo stato di funzionamento del satellite.

2.2.2 Segmento di Controllo Terrestre

Nel sistema sono presenti cinque stazioni terrestri che permettono il controllo delle orbite satellitari. Le stazioni sono dislocate lungo l'equatore, le basi sono alle Hawaii, Ascension Island, Diego Garcia, Kwajalein e Colorado Springs. Quest'ultima svolge la funzione di Master Station Control, ovvero di centro operativo del sistema.



Figura 2.3: La posizione delle stazioni terrestri

Il compito del segmento terrestre è quello di controllare il moto orbitale dei satelliti ed il funzionamento degli orologi di bordo. Inoltre deve azionare i reattori che vanno a modificare la posizione dei satelliti quando la loro posizione non corrisponde a quella assegnata, oppure disattivarli in caso di eventuali guasti.

Tutte le stazioni effettuano la funzione di monitoraggio (MS), ovvero ri-

cevano in maniera costante informazioni dai satelliti. I dati raccolti comprendono: il segnale di clock del satellite, i vari segnali di stato e le effemeridi dei satelliti. Le effemeridi informano il ricevitore sull'esatta posizione del satellite nello spazio, l'almanacco invece è un'informazione più generale che indica la posizione di tutti i satelliti della costellazione GPS. Ogni stazione, una volta raccolti i dati, li invia alla Master Station Control, attraverso queste informazioni la base di Colorado Springs effettua una valutazione quotidiana sia dell'orbita che dell'offset dell'orologio previsti per il giorno successivo su ciascun satellite. Successivamente queste informazioni vengono parametrizzate e inviate ai satelliti, i quali li comunicheranno agli utenti durante la giornata successiva.

Il corretto funzionamento del sistema GPS si basa su una stima molto precisa del tempo, perché ad un errore di un milionesimo di secondo corrisponde un errore di posizionamento di circa 300 metri. Alla luce di questo la MSC è dotata di una serie di orologi atomici che hanno la funzione di conservare il tempo del sistema al quale vengono riferiti tutti gli altri orologi atomici dei satelliti.

La funzione svolta dal segmento a terra è molto importante, infatti in assenza del controllo terrestre il sistema GPS risulterebbe non funzionante nel giro di 7-8 giorni.

2.2.3 Segmento Utente

Il terzo segmento è quello utente ed è rappresentato dal ricevitore di posizione dell'utente, che può essere portatile o fisso ed è munito di un' antenna ricevente. Esso può seguire solo uno o più codici dei satelliti che in quel momento sono ad esso visibili. Per questo motivo i ricevitori sono dotati di un'antenna con un'ottima sensibilità omnidirezionale. Visto che il sistema si basa sulla

misura del tempo, ogni ricevitore avrà un orologio, per questioni di costo meno preciso di quello dei satelliti, un software che decodifica i segnali ricevuti. Ogni satellite ha un solo nome denominato PNR, il nome è unico e può essere solo uno dei 32 consentiti nella costellazione. Ogni satellite trasmette a terra un messaggio, codificato tramite il suo *nome*; questo messaggio, chiamato almanacco, contiene i dati relativi alle orbite dei satelliti (effemeridi) ed altri dati caratterizzanti il satellite (tempo della settimana GPS, stato del satellite ecc.). Ogni ricevitore GPS ha al suo interno un comunissimo orologio al quarzo, i due orologi si sincronizzano riuscendo ad ottenere diverse informazioni:

- Il ritardo di sincronizzazione tra l'orologio del satellite e l'orologio del ricevitore GPS fornirà il tempo di percorrenza del segnale dall'antenna del satellite all'antenna del ricevitore GPS a terra.

Il funzionamento è abbastanza semplice, il ricevitore, che si trova in un punto a terra, misura la differenza di tempo tra t_1 , il tempo in cui è trasmesso il messaggio, e t_2 ovvero l'istante di tempo in cui esso è ricevuto dall'antenna.

Questo intervallo di tempo (t_2-t_1) viene poi moltiplicato per la velocità di propagazione delle onde nell'atmosfera, una velocità molto simile a quella della luce, in questo modo si riesce ad avere con un'approssimazione abbastanza precisa la distanza tra ogni satellite e l'antenna del ricevitore.

Nel caso in cui si volessero individuare tutte le tre dimensioni (quindi latitudine, longitudine ed altitudine) risulta necessario misurare i ritardi da almeno quattro satelliti.

L'accuratezza della rilevazione dipende da molteplici fattori, in maniera

particolare dal numero di satelliti visibili in quell'istante, dal tipo di codici utilizzati e da variazioni e ritardi dovuti ad effetti della ionosfera e troposfera.

- Se si conoscono i dati dell'orbita del satellite (e cioè si sa in maniera approssimata in quale punto dello spazio si trova il satellite), si può ricavare la posizione esatta dell'antenna del ricevitore GPS rispetto al centro di massa della Terra (tutti i satelliti orbitano intorno al centro di massa della Terra). Con una semplice operazione trigonometrica (compiuta dal software interno ad ogni ricevitore GPS) è possibile spostare il posizionamento dal centro alla superficie della Terra. Questa misura, riferita ad un solo satellite, fornisce un errore di posizionamento grossolano, pari a qualche centinaio di chilometro; se però viene ripetuta la misura su più satelliti (almeno tre per un posizionamento su latitudine e longitudine, almeno quattro se si vuol conoscere anche la quota del punto in cui ci si trova) si riesce ad ottenere la posizione con un errore di circa una decina di metri.

L'operazione di sincronizzazione degli orologi viene compiuta con i seguenti passi: il computer interno al ricevitore GPS genera di continuo delle copie dei nomi dei satelliti.

- Il segnale ricevuto dallo spazio viene captato dall'antenna del ricevitore GPS, ripulito e fatto passare in un blocco funzionale (interno al ricevitore GPS), chiamato auto-correlatore. L'auto-correlatore confronta i nomi dei satelliti generati dal computer interno del ricevitore GPS con i nomi dei satelliti captati dall'antenna del ricevitore stesso. Quando due dei nomi coincidono (sono costituiti dallo stesso blocco di codice) il computer del ricevitore GPS dice all'orologio interno di marcare il tem-

po. Il tempo marcato dall'orologio del ricevitore (il tempo impiegato dall'auto-correlatore per riconoscere che i due codici - quello satellitare e quello generato dal ricevitore - sono uguali) indica che è stata effettuata una sincronizzazione tra l'orologio del satellite e l'orologio del ricevitore ed equivale al tempo di percorrenza del segnale dall'antenna del satellite all'antenna del ricevitore a terra.

Le funzioni fondamentali di un ricevitore sono :

- Selezionare i quattro satelliti in funzione dei dati contenuti nell'almanacco
- Individuare i segnali dei satelliti decodificandoli attraverso il loro codice PNR
- Decodificare i dati di navigazione e memorizzarli
- Calcolare i tempi di arrivo dei segnali dei satelliti e quindi calcolare la distanza da essi
- Determinare la posizione del ricevitore e l'ora del sistema (GPS time)

2.3 Frequenze Utilizzate e codici trasmessi

Le due portanti sono modulate in fase e con diversi segnali :

- Codice P
- Codice C/A
- Messaggio di navigazione D

2.3.1 Codice P

Il codice P ha una frequenza di 10,23 Mbps, viene ripetuto periodicamente ogni circa 38 settimane e modula sia la portante L1 che L2, consentendo di ottenere la massima precisione possibile nel determinare la posizione terrestre. Tale codice viene utilizzato per eliminare, nell'utilizzo PPS, gli errori dovuti ai ritardi introdotti dalla ionosfera [Mum03]. Questo codice doveva essere utilizzato solo per scopi militari attraverso particolari ricevitori. Agli inizi degli anni '80 alcuni ricercatori riuscirono ad aggirare la protezione e per questo si è reso necessario l'utilizzo di un codice Y criptato.

2.3.2 Codice C/A

C/A sta per Coarse Acquisition (Acquisizione Grezza), infatti il segnale è studiato per usi civili, e quindi riserva il più alto grado di approssimazione (senza l'utilizzo del GPS differenziale). È impiegato nel posizionamento come il codice P. Questo codice fa parte dei Gold Code, codici pseudo casuali che hanno le caratteristiche di essere molto diversi l'uno dall'altro (si elimina quasi completamente la possibilità che il ricevitore confonda due satelliti) e di avere una bassissima correlazione con la propria copia tranne che in un

punto (si garantisce così l'esatta rilevazione del ToA, Time of Arrival, cioè del tempo impiegato al codice per arrivare a terra dal satellite).

Chipping rate (velocità di trasmissione dell'intero codice di 1023 bits) è di 1.023 Mbps, quindi per inviare un intero codice è necessario 1 microsecondo.

Quindi la frequenza con cui si succedono i bits di questo codice è 1.023 mhz. La distanza coperta dal codice nel tempo che intercorre tra una generazione e l'altra è il chip length, cioè la lunghezza d'onda del codice stesso. Il code length è la distanza coperta dal segnale nel tempo necessario a una ripetizione completa del codice. Tempo addietro si assumeva la precisione della misurazione pari all'1,2% della lunghezza d'onda del segnale con un errore conseguente di circa 3-6 metri mentre oggi, con il miglioramento delle tecnologie, si è arrivati a ridurre questo errore a 30-60 cm.

2.3.3 Navigation Data D

Il messaggio di navigazione viene trasmesso ad un basso bit/rate 50 Bps. L'intero messaggio, denominato frame è lungo 1500 BIT, quindi viene trasmesso in circa 30 secondi, in verità il frame è suddiviso in cinque sub-frame contenenti 300 BIT ciascuno. Il codice del messaggio è presente su entrambe le portanti e contiene :

- Le effemeridi del satellite, ovvero la sua posizione in un determinato instante ed alcuni parametri utili per il calcolo delle posizioni successive. Solitamente questi dati vengono aggiornati ogni ora.
- L'istante di trasmissione del messaggio
- Le correzioni per l'orologio di bordo
- Le correzioni per ritardi dovuti alla composizione dell'atmosfera.

- Lo stato dei vari satelliti, il cosiddetto Health
- L'almanacco ovvero le effemeridi, più o meno approssimate di tutti gli altri satelliti della costellazione.

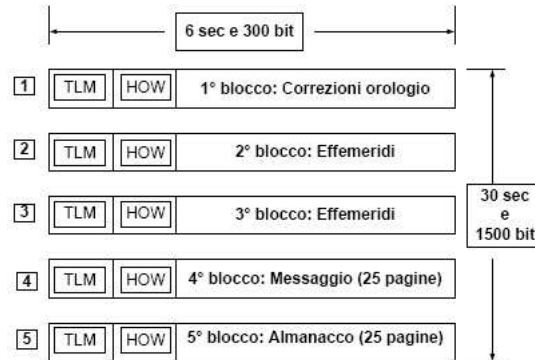


Figura 2.4: La struttura del Navigation Data Message

Ogni blocco viene trasmesso in 6 secondi, all'inizio di ciascun sub-frame sono presenti due codici comuni a tutti i blocchi, chiamati TLM (Telemetry World) e HOW (HandOverWorld).

TLM viene utilizzato dal sistema di controllo terrestre per verificare che la fase di aggiornamento sia avvenuta correttamente. HOW è un numero che moltiplicato per sei, fornisce il GPS TIME.

2.4 Fonti di errore

Il sistema GPS è soggetto a diversi tipi di errore che influiscono in maniera più o meno pesante sulla distanza misurata. Molti di essi possono essere risolti adottando delle formule correttive ottenute dopo un'attenta valutazione delle cause. Fondamentalmente essi sono:

- errori prodotti dalla propagazione dei segnali nell'atmosfera
- errori prodotti dalle percorsi multipath
- errori dovuti all'allineamento dei satelliti
- errori prodotti da ostacoli ambientali
- degradazione volontaria del segnale.

2.4.1 Errori dovuti alla propagazione dei segnali nella Troposfera e nella Ionosfera

Per troposfera si intende quella fascia di atmosfera a diretto contatto con la Terra, la cui estensione può variare dagli 8 Km dei poli fino ai 20 km all'equatore. La ionosfera va dai 60 fino a 450 Km di altezza e può essere divisa in strati differenti sia per composizione che per intensità di radiazione solare.

Nell'attraversare queste zone di atmosfera le onde dei satelliti subiscono sia variazioni di velocità, a causa della diversa composizione degli strati, che un effetto di rifrazione simile a quello che la luce solare subisce quando attraversa un blocco di vetro. Queste alterazioni influiscono negativamente sul sistema GPS allungando il percorso rispetto a quello previsto tra il satellite ed il ricevitore.

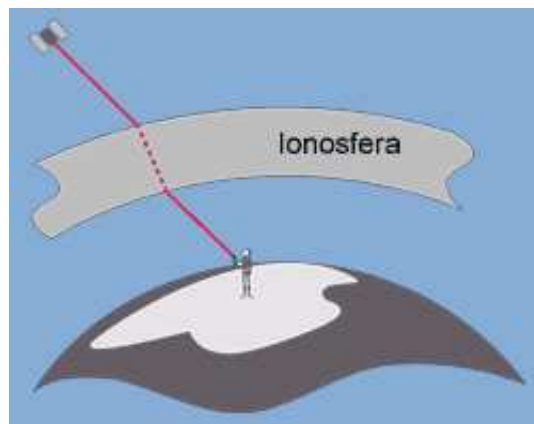


Figura 2.5: Errore dovuto alla variazione di velocità nella Ionosfera

Altro problema si verifica perché il segnale proveniente da satelliti a bassa elevazione potrebbe attraversare la Ionosfera per un periodo più lungo. Per eliminare questi tipi di disturbo sarebbe necessario l'utilizzo di entrambe le frequenze L1 ed L2, quindi anche del codice P che però non è disponibile per scopi civili.

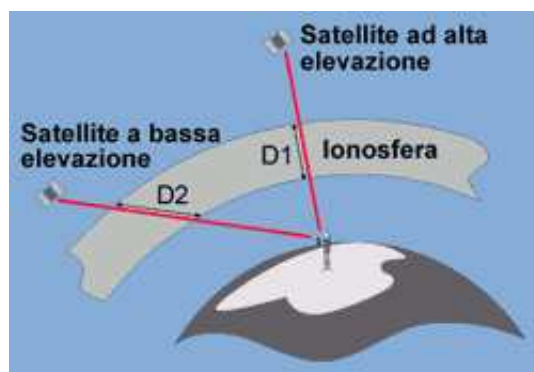
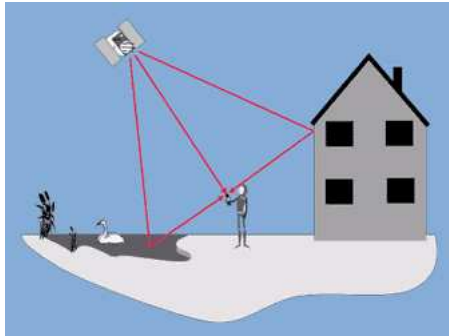


Figura 2.6: Il segnale ricevuto dai satelliti a bassa elevazione

2.4.2 Errori dovuti a percorsi multipath

Un errore di particolare rilevanza è quello dovuto al multipath. Esso si verifica quando le onde elettromagnetiche vengono riflesse da fabbricati, rocce o laghi. Il segnale infatti viene riflesso da questi ostacoli generando false misurazioni. Le antenne di ultima generazione sono in grado di filtrare il fenomeno del multipath, se si vuole ottenere però una massima accuratezza esistono delle particolari antenne denominate choke ring che eliminano a priori tutti i segnali indiretti.



(a) Il fenomeno del multipath



(b) Antenna Choke Ring

Figura 2.7: Errori causati dal multipath

2.4.3 Errore della posizione nello spazio dei satelliti

Si possono verificare situazioni in cui i satelliti visibili sono eccessivamente ravvicinati tra loro. In figura viene mostrato un caso di buona geometria ed uno di pessima.

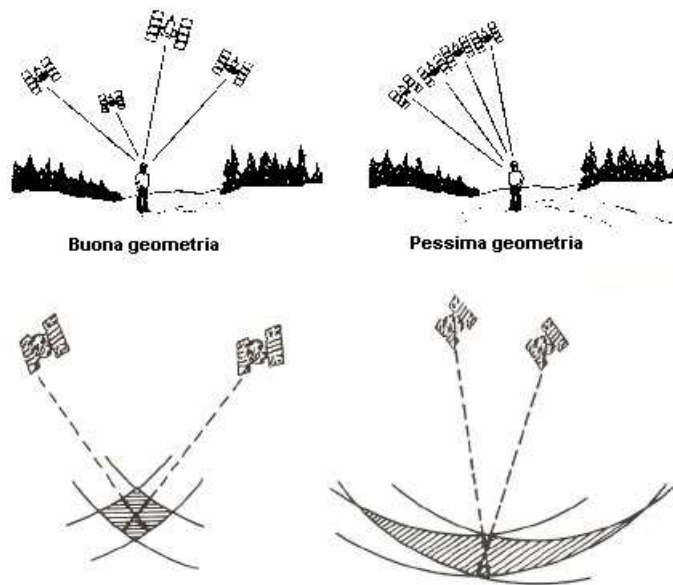


Figura 2.8: Buona e pessima geometria dei satelliti

In condizioni pessime l'area tratteggiata (ovvero l'area in cui sicuramente ricade il punto di osservazione dati i segnali di due satelliti) aumenta, in questo modo aumenta anche il grado di incertezza del sistema.

2.4.4 Errori dovuti a ostacoli ambientali

Il segnale GPS viene in qualche modo schermato da materiali come metalli e cemento, che molto spesso rappresentano un ostacolo insormontabile. Fortunatamente materie come la plastica, la gomma e la stoffa non degradano in maniera importante il segnale. La vegetazione invece si trova in uno stato

intermedio, solitamente il fogliame seppur non troppo alto e con un elevato tasso di umidità non costituisce un grosso ostacolo.

Altro problema che riscontriamo in molte zone del centro cittadino di Bologna, così come in molti centri storici delle più grandi città italiane, è quello di avere strade particolarmente strette con abitazioni alte, questo fa sì che non tutti i satelliti siano visibili.

2.4.5 Degradazione volontaria del segnale

Fino al maggio del 2000, veniva effettuata una degradazione del segnale, quindi il grado di precisione era nell'ordine di 100-150 metri. Con il decreto del presidente degli USA Bill Clinton, venne abolita la degradazione del segnale; in questo modo si riesce attualmente ad ottenere una precisione della posizione rilevata pari a 10-20 metri.

2.5 Galileo

Galileo è un sistema di posizionamento finanziato in parte dall'Unione Europea che ha come scopo principale quello di essere non soltanto un'alternativa al GPS americano, ma di migliorare notevolmente la precisione della rilevazione e di introdurre caratteristiche esclusive. Esso si inserisce in un programma molto più grande che ha preso avvio con EGNOS [Wik07b]. Quest'ultimo è stato sviluppato concretamente a partire dal 1999 ed in Europa il segnale è risultato disponibile dal 2001.

Il programma GALILEO fu avviato nel maggio del 2003, attraverso un accordo con l'Unione Europea e l'Agenzia Spaziale Europea. Alla base di una scelta molto forte come quella di avviare un progetto per il posizionamento satellitare risiede la convinzione che in un futuro non molto remoto la società moderna dipenderà sempre più dall'uso di applicazioni GNSS, sviluppate sia per attività economiche ma anche per aspetti di sicurezza vitali. Alla luce di queste considerazioni non si è più disposti a dipendere per un periodo di tempo medio lungo da segnali GNSS esteri sulla cui qualità, disponibilità e prezzo non si ha praticamente nessun controllo.

Inoltre si ipotizzano ripercussioni economiche molto importanti su svariati mercati, questi presupposti giustificano il completamento del programma Galileo come un investimento infrastrutturale indispensabile per l'EU [Muc06].

L'entrata in vigore del servizio è prevista per il 2011, ma per problemi economici/politici potrebbe slittare al 2014. Galileo potrà contare su 30 satelliti orbitanti su tre piani inclinati all'equatore, di questi 27 saranno utilizzati effettivamente ed altri 3 saranno disponibili in caso di necessità come per esempio per un malfunzionamento di uno dei 27.

In questo modo l'EU cerca di distaccarsi completamente dal sistema sta-

tunitense, che pur essendo attualmente del tutto gratuito, è un sistema nato per scopi militari e solo successivamente adattato ad un utilizzo civile. Inoltre non viene garantito in nessun modo il funzionamento del sistema, si potrebbero quindi verificare situazioni in cui volutamente alcune aree risultino tagliate fuori dalla ricezione del segnale. Il Dipartimento della Difesa degli Stati Uniti d'America si è sempre riservata il diritto di ridurre la copertura del segnale e la precisione dello stesso in qualsiasi momento.

Altro motivo che spinge l'Unione Europea a volere un programma di questa portata, è l'opportunità di profitto legato ai servizi e ai dispositivi associati al posizionamento satellitare. Si prevede che nei prossimi anni il 10% dei veicoli possa essere collegato alla rete Internet, questo porterebbe enormi vantaggi non solo per i singoli automobilisti ma anche per gli Stati stessi, che avrebbero sia efficaci sistemi di gestione che di monitoraggio del traffico, riuscendo in parte a sopperire alle carenze infrastrutturali.

Il sistema GALILEO sarà completamente compatibile ed inter operante con il GPS, nel giugno del 2004 si è arrivati ad un accordo con gli USA per utilizzare uno schema di modulazione denominato Binary Offset Carrier 1.1 che permette la trasmissione dei servizi sulle stesse frequenze del GPS, senza però ostacolarne il funzionamento e la ricezione dovuti a possibili interferenze.

Inoltre il segnale verrà trasmesso su almeno due frequenze diverse: in questo modo, il Ground Control System sarà in grado di correggere in maniera più accurata i disturbi atmosferici e calolerà la posizione con una precisione di circa un metro, invece dei quasi venti metri del sistema statunitense.

Il lancio in orbita di alcuni satelliti di test era previsto per il 2007-2008, ma ultimamente problemi di natura sia finanziaria che politica, hanno messo in discussione anche l'effettiva necessità di un progetto così costoso. Infatti tutti gli Stati appartenenti all'EU fanno parte dell' ESA, inoltre vi è la coesistenza

di diversi paesi al di fuori dell'EU come la Cina, Israele ed altri la cui posizione non risulta essere molto chiara.

Da un punto di vista tecnico la Russia sta pensando di integrare GALILEO con il proprio sistema di posizionamento denominato GLONASS.

Nel Dicembre del 2005 è avvenuto il lancio del satellite denominato GIOVE A ed è previsto il lancio di altri due satelliti GSTB-V2/A e GSTB-V2/B la cui sigla sta per Galileo Satellite Test Beds che permetteranno di verificare il funzionamento del sistema e la stabilità degli orologi in orbita. Questi nuovi satelliti saranno equipaggiati con un orologio atomico all'idrogeno oltre a quello in rubidio. Successivamente verranno mandati in orbita altri due satelliti che completeranno la fase di verifica e convalida.

Caratteristica fondamentale di Galileo è la possibilità di gestire una vasta gamma di velocità di trasmissioni dati da un minimo di 250 bit/s fino ad un massimo di 1500 bit/s.

Si moltiplica quindi la possibilità di impiego dei messaggi e in base a tali funzionalità potrebbero essere costruiti molti servizi.

Il programma Galileo è stato diviso in tre parti distinte:

1. Definizione
2. Sviluppo e validazione in orbita
3. Sviluppo completo ed operazioni

La prima fase è stata completata nel 2003. Quella di validazione sarà ultimata al lancio di tutti i satelliti di test, che riescono ad implementare in modo pienamente rappresentativo la costellazione dei 30 satelliti. Solo quando la fase di test sarà ultimata e i satelliti saranno completamente funzionanti si passerà al lancio dei 26 mancanti.

La geometria della costellazione è stata studiata in modo tale che l'orbita permetta ai pannelli solari di essere sempre esposti ai raggi del sole.

2.5.1 Servizi Offerti

Svariati sono i vantaggi che GALILEO potrà offrire al pubblico, in particolare i settori che potrebbero essere stimolati sono :

- Ambiente, per minori ingorghi stradali e itinerari più corti che ridurrebbero i consumi.
- Occupazione
- Vantaggi sociali, per una maggiore sicurezza
- Servizi Pubblici, per servizi di ricerca e soccorso, di autoambulanza etc...
- Aviazione.

Una caratteristica distintiva rispetto al GPS è quella di offrire la funzionalità di posizionamento all'interno degli edifici, cosa attualmente non disponibile con il sistema americano.

Con esso oltre a distaccarsi dalla dipendenza di un solo sistema, si utilizza uno strumento in cui sono ben definite, fin dall'inizio del progetto, responsabilità chiare e non ambigue. Inoltre vengono distinte le funzionalità per operatori pubblici o privati.

I servizi offerti si possono riassumere in quattro grandi categorie:

1. Un servizio di base (Open Service), disponibile in modo gratuito ed equiparabile a grandi linee al servizio offerto dal GPS ma con qualità ed affidabilità maggiori.

2. Un servizio commerciale (Commercial Service), si tratta di uno strumento a pagamento accessibile mediante un canale cifrato per tutti coloro che richiedono prestazioni di navigazione superiore e servizi addizionali.
3. Un servizio vitale (Safety Life), di altissima qualità il cui presupposto è salvaguardare la vita umana, destinato al settore aereo e marittimo. Questo servizio sarà distribuito in maniera gratuita ma richiederà l'uso di ricevitori più sofisticati e certificati.
4. Un servizio Governativo riservato in maniera primaria alle necessità della pubblica amministrazione in materia di protezione civile, sicurezza nazionale e rispetto del diritto.

La chiara definizione degli aspetti di certificazione, standardizzazione e legali, determineranno il successo o l'insuccesso di GALILEO.

I servizi di GALILEO copriranno tutta la Terra, tuttavia in alcune zone sarà possibile collocare le cosiddette LOCAL ELEMENT per migliorare ulteriormente il grado di precisione ed affidabilità del segnale.

Un LOCAL ELEMENT è sostanzialmente un'infrastruttura fissa utilizzata da un Service Provider per soddisfare determinate necessità.

La vera novità presente in GALILEO non sarà tanto legata ad una maggiore precisione della rilevazione, bensì alla possibilità di integrazione con la comunicazione mobile sia essa wireless che satellitare.

Tali applicazioni prendono il nome di Location Based Services (LBS), e saranno disponibili su cellulari o PDA che integrano alcune funzioni di navigazione.

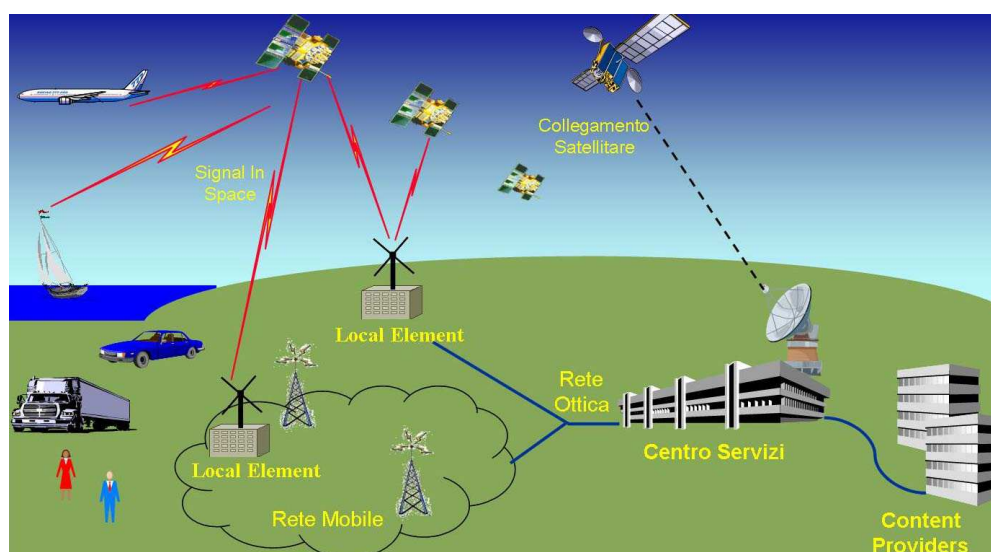


Figura 2.9: I componenti del Sistema GALILEO

Quindi lo scenario che si va prospettando è quello che un utente richieda a GALILEO la sua posizione e riceva da un Centro Servizi, attraverso una comunicazione mobile, informazioni contestuali alla sua posizione.

Il Centro Servizi ha il compito di gestire ed integrare le informazioni di posizione con applicazioni a valore aggiunto.

Le principali categorie di servizi di una LBS sono:

- Informazioni generali e di navigazione, in questa categoria troviamo aspetti simili alle pagine gialle, servizi di informazione turistica, e servizi di utilità generale, quali informazioni meteo o ottimizzazione del percorso in funzione delle situazioni attuali di traffico;
- Servizi di Assistenza, in questo modo verranno localizzati immediatamente i cittadini che chiamano il servizio E-112 in caso di emergenza;
- Servizi di tracking, per una o più persone;
- Servizi di rete, si prospetta un trend sempre crescente per il Mobile

Advertising, cioè la possibilità di spedire al terminale offerte pubblicitarie *ad hoc* in funzione della posizione e quindi del luogo in cui si trova l'utente.

Molti settori beneficeranno del sistema GALILEO, tra questi vi sono il settore dei trasporti, quello ferroviario, dell'aviazione e i trasporti marittimi. La comunicazione tra i terminali avverrà tramite diversi protocolli e fornirà un insieme di servizi oggi conosciuto come Advanced Driver Assistance System (ADAS) che prevede un'integrazione con i sottosistemi di bordo e di navigazione per migliorare la mobilità.

L'uso combinato di navigazione e comunicazione per tutte le applicazioni che richiederanno l'invio a distanza da e verso un terminale, sarà determinante per ottenere informazioni ad elevato valore aggiunto.

Capitolo 3

Le Reti Wireless

In questo capitolo vengono introdotte le reti wireless, in particolare modo vengono messe in evidenza le caratteristiche distintive delle reti MANET. Successivamente vengono discusse le reti veicolari VANET, come un'estensione di una rete multi hop, vengono menzionati alcuni possibili soluzioni per la comunicazione in broadcast tra i nodi della rete in uno scenario che risulta altamente complesso e difficilmente prevedibile a priori.

3.1 Reti Wireless

Le comunicazioni wireless, ed in particolare il Wireless Networking, rappresentano una tecnologia in rapida espansione che consente l'accesso a reti e servizi senza necessità di cablaggi. In via teorica, gli utenti di una rete locale wireless vogliono usufruire degli stessi servizi e vorranno disporre delle stesse potenzialità a cui una rete cablata li ha abituati. In pratica, l'equivalenza tra i due approcci wireless e wired, è una sfida aperta. In particolare l'approccio wireless, a fronte di innegabili vantaggi, è soggetto ad alcuni limiti non presenti nell'approccio cablato.

3.1.1 Interferenza e Affidabilità

L'interferenza nelle comunicazioni wireless può essere causata dalle cosiddette collisioni, ovvero trasmissioni simultanee da parte di due o più terminali wireless nella stessa banda di frequenza. In realtà il problema dell'interferenza è più ampio e coinvolge anche dispositivi di uso comune che non hanno nulla a che vedere con le wireless LAN ma che possono causare non pochi problemi al funzionamento di queste ultime (ad esempio, un comune forno a microonde che opera nella banda 2.4-2.5 GHz).

3.1.2 CSMA/CA

Nelle reti cablate la tecnica usata per la comunicazione è la CSMA-CD (Carrier Sense Multiple Access-Collision Detected), nessun host può iniziare una trasmissione se il canale occupato da un'altra trasmissione.

In reti wired, se un host è in fase di trasmissione, tutti gli altri host sono in grado di ascoltare la trasmissione quindi è possibile regolare la trasmissione con la tecnica delle attese o back-off.

Il CSMA/CA è derivato dal CSMA/CD, il principio base del CSMA/CA è sempre quello di ascoltare prima di trasmettere, cosa che però non impedisce le collisioni. Si tratta di un meccanismo di base asincrono (senza connessione) che fornisce il così detto best effort service (fa cioè del suo meglio per soddisfare una richiesta) ma non dà nessuna garanzia di velocità e di latenza. Il suo vantaggio principale è che è adatto a supportare la trasmissione dei datagram IP e che si adatta bene alle condizioni variabili di traffico. Inoltre è molto robusto contro le interferenze.

3.1.3 Terminale nascosto

Si dice che un nodo 1 è nascosto ad un nodo 3 appartenente alla stessa BSS, quando i due nodi sono fuori dalla reciproca portata radio, cioè non possono comunicare direttamente.

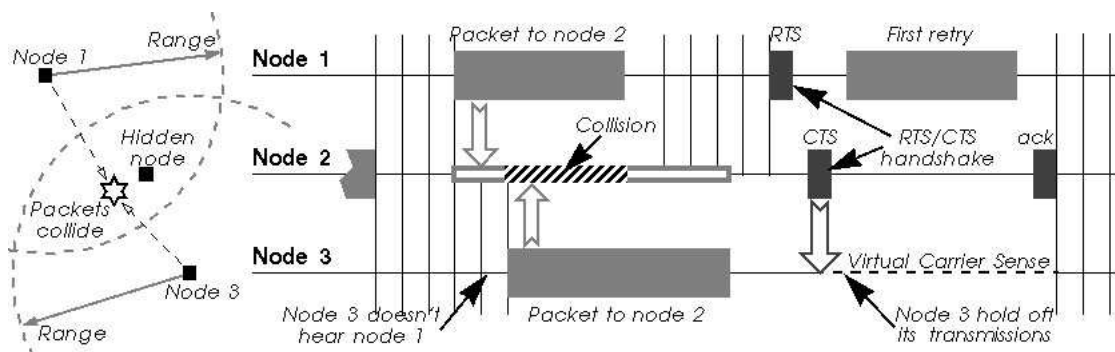


Figura 3.1: Terminale nascosto RTS/CTS

Una soluzione semplice ed elegante a questo problema è l'utilizzo del meccanismo RTS/CTS (Request To Send/Clear To Send). Quest'ultimo è semplicemente un handshaking: prima di inviare il pacchetto il mittente inoltra un RTS aspettando un CTS dal receiver. La ricezione di un CTS indica che il canale è libero, il frame di dati sarà inviato esattamente un SIFS dopo la trasmissione del frame CTS, senza controllare il mezzo trasmissivo.

Allo stesso tempo, ogni nodo che si trova nel raggio di trasmissione vedendo transitare un CTS capisce che è in corso una trasmissione dati. I nodi che ascoltano un CTS sono terminali che potrebbero creare una collisione. Inoltre nei pacchetti RTS /CTS è indicata la lunghezza potenziale della trasmissione (campo NAV). In questo modo si riesce ad ottenere una buona stima del tempo necessario per effettuare la trasmissione. Proprio per quest'ultima caratteristica il meccanismo RTS/CTS è anche conosciuto come virtual carrier sense.

Il meccanismo RTS/CTS presenta anche altri vantaggi:

- Abbassa notevolmente l'overhead causato dalle collisioni sul mezzo trasmissivo.
- Nel caso in cui due terminali desiderano trasmettere nello stesso istante, i loro RTS collidono e nessuno dei due riceve un CTS, così facendo viene perso solo un pacchetto RTS, in uno scenario normale avrebbero perso l'intero pacchetto dati.

Per contro l' RTS/CTS handshaking aggiunge un significativo overhead, per questo solitamente non viene utilizzato per inoltrare tutti i pacchetti. Solamente i frame di dimensione maggiore del parametro `RTSThreshold`, saranno soggetti al meccanismo RTS/CTS .

La trasmissione broadcast (o multicast) di frame dati, in generale non è soggetta al meccanismo RTS/CTS. Questo è vero però solo per i frame in cui il bit `ToDS` è pari a 0, nel qual caso viene usata la procedura di accesso base e non viene inviato alcun frame ACK.

Nel caso in cui un frame broadcast (o multicast) abbia invece il bit `ToDS` pari a 1, tale frame segue le regole della procedura di accesso con RTS/CTS, questo perchè il suddetto frame è diretto verso l'AP. La suddetta distinzione in base al valore del bit `ToDS` si riflette anche sulla ritrasmissione dei frame: solo nel caso in cui `ToDS` sia pari a 1 è previsto il meccanismo di recupero dei dati non ricevuti correttamente (ovvero la ritrasmissione dei frame).

3.2 Reti MANET

Le MANET (Mobile Ad-hoc Network) costituiscono un sistema di nodi dotati di mobilità e capacità di organizzazione dinamica in reti o sottoreti con topologie arbitrarie [CM99].

Una rete ad hoc mobile è anche definita come un sistema autonomo di router mobili connessi mediante collegamenti wireless. Tutti i nodi del sistema collaborano svolgendo operazioni di instradamento.

Nelle reti MANET la conoscenza delle rete non è data a priori, quindi deve essere scoperta attraverso la comunicazione tra i stessi nodi. Col passare del tempo ogni nodo ha una visione sempre più dettagliata sulla rete ed i possibili percorsi esistenti.

Il traffico di rete nelle MANET denota alcune differenze rispetto a quello esistente in WLAN con un infrastruttura di rete:

- Alta mobilità dei nodi
- Assenza del controllo e della gestione della rete.

3.2.1 Alta mobilità dei nodi

A causa della mobilità imprevedibile dei nodi, la topologia di rete può cambiare costantemente. Le reti ad hoc vengono costruite all'occorrenza ed utilizzate in ambienti estremamente dinamici, i nodi infatti possono muoversi liberamente e dinamicamente auto-organizzandosi e creare una topologia di rete arbitraria senza l'aiuto di un'infrastruttura preesistente.

3.2.2 Assenza del controllo e della gestione della rete.

Data l'assenza di infrastrutture fisse, nelle MANET la gestione ed il controllo della rete è distribuita tra i vari nodi, i quali, devono farsi carico sia del routing che della sicurezza della comunicazione.

Sostanzialmente gli algoritmi di routing devono :

- Assicurarsi che le tabelle di routing siano ragionevolmente piccole, anche alla luce delle risorse ridotte delle quali spesso dispongono i nodi in una rete ad hoc [GS];
- Riuscire a selezionare il miglior percorso possibile per inoltrare un pacchetto da A a B;
- Mantenere le tabelle di routing aggiornate il più frequentemente possibile perché la topologia di rete potrebbe cambiare in maniera molto veloce [AE03];
- Arrivare ad un funzionamento ottimo in un piccolo intervallo di tempo inoltrando il minor numero possibile di pacchetti.

In qualsiasi rete MANET il routing rimane uno dei maggiori problemi. Questa difficoltà è data principalmente dal fatto che tutti i nodi sono mobili. I protocolli di routing possono essere classificati in tre grandi categorie:

- Protocolli proattivi effettuano un broadcast ad intervalli regolari per aggiornare la tabella di routing, tra questi troviamo DSDV, OLSR, FSR, TBRPF.
- Protocolli reattivi, che utilizzano un path discovery On-demand, possiamo prendere di esempio AODV e DSR [SPKW07].

- Protocolli ibridi (reactive and proactive) in cui possono essere inseriti ZRP e LANMAR.

3.3 Reti VANET

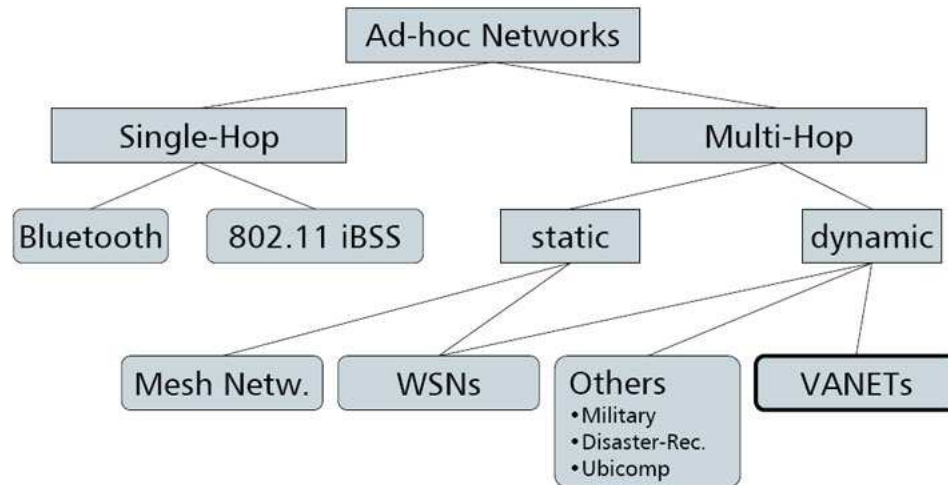


Figura 3.2: Ad Hoc NetWorks

Le reti VANET ovvero ‘Vehicular ad-Hoc network’ dette anche IVC sono una sotto categoria delle reti MANET. Esse sono formate da host che potrebbero essere mobili, quindi i dispositivi non usano necessariamente una infrastruttura di rete pre-esistente. I problemi di queste reti ad alta ‘mobilità’ consistono nella fase di route discovery e dal fatto che un buon cammino trovato dopo pochi istanti non potrebbe più essere valido perché un host ha abbandonato la rete.

Le differenze principali tra le reti VANET e le quelle MANET sono la velocità di spostamento dei nodi e di conseguenza il rapido mutamento della topologia delle reti e delle sotto reti [Pol06]. La comunicazione tra i veicoli stessi o quella che avviene tra i nodi ed un server centrale diviene una priorità importante. In questo modo è possibile:

- Segnalate situazioni di pericolo in strade ed autostrade in caso di incidenti attraverso la diffusione di messaggi di warning;

- Fornire segnali di sicurezza attiva: per esempio in situazioni meteorologiche avverse;
- Indicare l'attuale velocità media di una determinata strada;
- Consigliare percorsi alternativi.

Quindi la tecnologia VANET se efficientemente implementata, può diventare uno strumento molto utile per la sicurezza stradale.

È molto importante sottolineare che le reti VANET hanno caratteristiche peculiari rispetto le generiche MANET. I comportamenti dei nodi (in termini di velocità e di abbandono del network) sono differenti in funzione dello scenario in cui si trovano, per esempio ci sono significative differenze tra una strada urbana ed una autostrada o statale.

Inoltre molti dei problemi esistenti per reti ad-hoc generiche non si presentano in quelle veicolari:

- Possibilità maggiore di impiego di risorse hardware. I dispositivi infatti sono alimentati tramite la batteria della vettura, in questo modo lo sforzo degli algoritmi può essere focalizzato sulle performance e non sul consumo energetico.
- Possibilità di immagazzinare i dati in dispositivi di massa sia lato client sia su un eventuale server a cui saranno inviate informazioni.
- I nodi, attraverso l'uso di sistemi di posizionamento GPS e grazie all'uso di mappe dettagliate dei territori, hanno una buona conoscenza del territorio circostante.

Molti sono i campi di ricerca ancora aperti, essi spaziano dalla realizzazione dello scheduling broadcasting, compito in ogni caso molto complesso

data l'assenza di informazioni sulla topologia di rete, sino ad arrivare alle problematiche che comporta l'impiego del flooding in un protocollo senza rilevamento delle collisioni (CSMA-CA).

3.3.1 Protocolli di Routing

I classici protocolli di routing utilizzati nelle MANET o un protocollo di broadcast, difficilmente possono essere usati in contesti VANET, perché necessitano prima di scoprire la rotta e successivamente costruire una lista di nodi vicini. Questi compiti richiedono del tempo per essere effettuati e non sono quindi adatti per disseminare informazioni inattese come messaggi di allarme in caso incidente. D'altro canto un classico protocollo di broadcast non può essere usato proprio perché potrebbe causare sia un overhead sia un alto numero di collisioni [EF05].

Procolli basati sulla localizzazione

I protocolli basati sulla localizzazione usano un'informazione addizionale, ovvero la posizione geografica del nodo, per effettuare la fase di routing. In questo modo si cerca di superare le limitazioni di protocolli topology based. La localizzazione è usata dal sender per localizzare il receiver. La posizione è inclusa nel campo di *address destination* del pacchetto, che sarà usato nel realizzare il routing. Le decisioni sul percorso di routing dipendono solo sulla posizione del destinatario e dalla posizione dei nodi vicini più vicini.

In questo modo tali protocolli non necessitano di stabilire o mantenere le rotte, immagazzinare dati nella tabella di routing, nè scambiare messaggi di controllo. Inoltre questi protocolli consentono di mandare messaggi di broadcast verso una data area geografica in un modo naturale (geo-cast).

In questa categoria di protocolli si possono distinguere:

- LAR che è un protocollo on-demand [KV00]
- DREAM che è un protocollo proattivo
- GPSR che usa le informazioni di localizzazione dei vicini, ricevute in maniera periodica, per inoltrare i pacchetti.

Nelle reti VANET un veicolo inoltra in broadcast un messaggio di allarme a tutti i suoi vicini, senza avere nessuna informazione sulle loro identità. Il broadcast, che è considerato un'operazione base, è previsto che sia eseguito frequentemente nelle MANET.

In effetti, esso è utilizzato dai protocolli di routing sia per effettuare la fase di discovery path che per costruire e mantenere le proprie tabelle di instradamento. Esso è anche utilizzato per il multicast nelle reti mobili per disseminare messaggi di allarme nel caso incorrono casi di emergenza.

Proprio per la mancanza di informazioni sulla topologia di rete, lo scheduling del broadcast è un compito abbastanza difficile. Una soluzione molto semplice sarebbe quella del flooding.

Ogni veicolo che riceve un messaggio, immediatamente effettua un broadcast dello stesso a tutti i suoi vicini e così via, fintanto che il broadcast non è ripetuto per l'intero network.

In ogni caso, questa soluzione presenta diversi problemi, quali:

- Contesa del mezzo trasmissivo
- Collisioni
- Eccessiva ridondanza dei messaggi di broadcast.

L'insieme di queste complicazioni è conosciuto sotto il nome di 'storm broadcast problem'.

Per ridurre la ridondanza del re-broadcast, alcune euristiche sono proposte:

- random generation
- approccio basato sull'uso dei contatori
- approccio basato sulla localizzazione
- approccio basato sul concetto di cluster.

Una nuova proposta denominata RBM (Role-Based Multicast), riduce il numero di ridondanza. Il veicolo (nodo) che inoltra il re-broadcast deve assicurare la presenza di nodi vicini che possano ricevere il messaggio. Sostanzialmente se non esiste un vicino, viene detto al nodo di non ri-inviare il messaggio. In questo modo il numero di ripetizioni del messaggio si riduce notevolmente. In ogni caso questo metodo richiede che ogni nodo mantenga la lista di tutti i suoi vicini. La necessità di mantenere una lista dei nodi più prossimi e la fase di detection, genera un significativo overhead nel network.

Poiché i messaggi di allarme sono inaspettati, la determinazione del set di nodi vicini, aumenta il tempo di ritrasmissione. Inoltre RBM non supera il problema della frammentazione e non tiene conto della direzione di circolazione all'interno della strada. In base ai risultati che si possono vedere nelle simulazioni effettuate, questo protocollo collassa quando la percentuale di veicoli equipaggiati con questo sistema è compresa tra il 5% e 25% [WXG06].

Sono stati proposti altri schemi, TRADE e DDT, per migliorare il broadcast nelle reti VANET.

In TRADE ogni veicolo deve effettuare un broadcast per determinare la posizione e la direzione di circolazione dei nodi ad esso più prossimi. Successivamente ogni veicolo indica il nodo che ritrasmetterà il messaggio, tutto

questo evita la ridondanza ed il re-broadcast. In ogni caso, questa soluzione richiede di mantenere il set dei vicini su tutti i veicoli, quindi risulta essere molto dispendiosa.

Per superare questo problema è stato proposto uno schema basato sulla localizzazione GPS chiamato DDT. Il vero problema con il DDT è che un veicolo effettua l'operazione di rebroadcast solo una volta.

Tale protocollo risulta essere inutilizzabile in una strade non molto trafficate o in situazioni in cui i veicoli sono distanti. I risultati di alcuni test effettuati, dimostrano che la riusabilità del DDT aumenta in maniera proporzionale al raggio di trasmissione. In ogni caso una perfetta riusabilità non è ottenuta perfino con un range di trasmissione uguale a 2000 metri. In DDT, alcuni veicoli non possono essere informati, rappresentando un pericolo in caso di emergenza.

ODAM

In O DAM, viene generalizzato il concetto del DDT e vengono superati i problemi precedentemente presentati:

- Frammentazione
- Riusabilità
- Determinazione dei vicini.

Per fare questo, sono stati effettuati dei rebroadcast periodici dei messaggi di allarme, ed è stato introdotto un tempo di relay dinamico. Sostanzialmente il tempo di relay è assegnato in funzione della distanza dal sender. Test dimostrano che questo protocollo risulta molto più potente degli altri metodi RBM, TRADE e DDT. Inoltre dai risultati si può evincere che il protocollo risulta efficace in qualsiasi condizione di traffico.

ODAM [Ben04] è un protocollo usato per propagare messaggi di allarme all'interno di reti veicolari, allo scopo di prevenire il tamponamento in caso di nebbia, incidenti o altri ostacoli.

Esso introduce un'efficiente disseminazione dei messaggi di allarme, restringendo il rebroadcast solo a nodi speciali, chiamati relay, ed in zone ristrette, denominate risk zone.

La scelta di usare ad hoc network, invece di cellular network, è giustificata dal fatto che non si necessita di un'infrastruttura di rete. Questo evita in molti casi il problema di disponibilità del network stesso, inoltre la totale assenza di un'infrastruttura e di apparati di rete come router o multiplexes, consente uno sviluppo rapido ed economico di tali network.

Per risolvere il problema della contesa del mezzo trasmissivo, il messaggio non viene ritrasmesso immediatamente, ma il re-broadcast è ritardato per un lasso di tempo chiamato defertime.

In questo modo il numero di re-broadcast simultanei ed il numero di collisioni si riducono. Il valore del defertime viene generato in maniera random o calcolato in funzione della distanza dei nodi.

Ogni veicolo che utilizza ODAM dovrebbe essere equipaggiato con un device abilitato per ottenere le sue coordinate geografiche al tempo corrente. Il GPS riesce a fornire coordinate in 3D ma per semplicità sono state usate solo le coordinate x e y.

La comunicazione tra i veicoli si suppone essere bi-direzionale ed è basata su messaggi di broadcast. Ogni veicolo ha un identificativo unico denominato *node_id*.

In un futuro non molto prossimo è probabile che larghe zone saranno coperte da Internet (per esempio attraverso Wi Max) e quindi i veicoli potranno collegarsi alla rete Internet ed eseguire applicazioni multimediali. In

ODAM si è preferito utilizzare lo schema del pacchetto IP, ogni messaggio di broadcast deve avere sia un identificativo unico (`node_id`) che indica il veicolo sia un `message_id` che indica il numero di sequenza del messaggio.

Non è necessario specificare l'indirizzo di destinazione, questo consente di ridurre l'overhead introdotto nell'intestazione del pacchetto. Entrambe le versioni dell'IPv possono essere utilizzate, anche se gli sviluppatori preferiscono IPv6.

Quando accade un incidente, il veicolo danneggiato o qualsiasi altro veicolo che scopra tale problema, devono inoltrare un messaggio di allarme per informare gli altri nodi su quanto è accaduto.

Una volta che il messaggio viene inoltrato solo le vetture che sono nella risk zone riescono a catturarlo. Tra tutti i vicini della stessa zona, solo un veicolo, denominato relay, deve essere eletto per assicurare il re-broadcast del messaggio di allarme per informare tutti i veicoli che ancora non lo hanno ricevuto.

Il veicolo di relay è selezionato in maniera distribuita. Ogni veicolo può sapere, dall'informazione presente nel messaggio di allarme, se diventerà un relay o meno. Il relay deve essere selezionato in modo da assicurare la copertura di una zona più grande possibile tra quelle non ancora coperte dal sender. Quindi il relay deve essere il vicino più lontano dal sender.

Il veicolo designato ad essere il relay è quello che ha il minimo valore di defertime computato. Il nodo che riceve un messaggio di allarme non dovrebbe fare un re-broadcast immediatamente, ma aspettare il periodo di defertime. Il valore di defertime è inversamente proporzionale alla distanza che intercorre tra il sender ed il reciver.

Alla scadenza di questo periodo, se un nodo non riceve un altro messaggio di allarme con lo stesso `message_id`, allora effettua il re-broadcast del messag-

gio. In questo modo esso viene eletto come relay. Per avvantaggiare il veicolo più lontano dal sender a divenire relay, il deferttime deve essere inversamente proporzionale alla distanza che lo separa dal sender.

$$defertime(x) = \max_defer_time \frac{(R^r - D_k)}{R^r}$$

Figura 3.3: Formula del Defertime

Se si suppone che la distribuzione dei veicoli sia uniforme, si può scegliere $E=2$. Per ogni messaggio ricevuto, il veicolo deve indicare la sua posizione rispetto al nodo incidentato.

Il comportamento di ODAM è simile a quello di una macchina a stati finiti. Inizialmente tutti i veicoli sono nello stato E0. Quando un veicolo incorre in un incidente si sposta nello stato E1, da questo istante inoltra messaggi di allarme periodicamente.

Tutti i veicoli che ricevono il messaggio si spostano nello stato E2, aspettano il tempo di defertime assegnato. Il veicolo con il defertime minore si muove nello stato E3 diventando relay. Tutti i veicoli che sono nello stato E2 si muovono nello stato E4, il re-broadcast del messaggio è infatti assicurato dal relay.

In ogni caso il re-broadcast è effettuato solo da veicoli che sono all'interno della risk zone, in questo modo viene evitato il problema di un infinito re-broadcast.

Una macchina a stati finiti viene assegnata ad ogni messaggio di allarme, ogni veicolo infatti mantiene una tabella definita t_state che è costituita da una lista con i seguenti valori: $(msg_key, state)$. Il msg_key è una coppia di valori $(node_id, message_id)$, mentre lo state indica lo stato del veicolo.

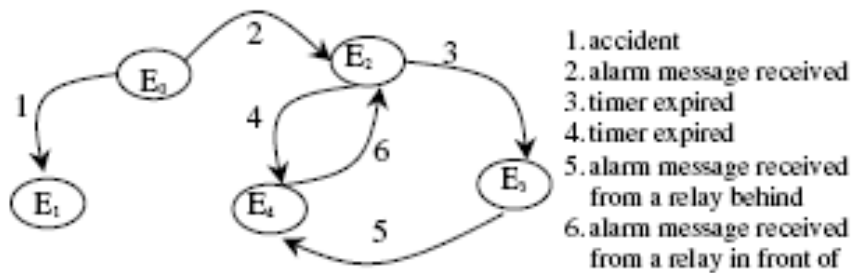


Figura 3.4: Gli stati dei messaggi di allarme

Per ogni messaggio di allarme, lo stato 0 rappresenta sia lo stato iniziale che quello finale. Nel caso in cui i nodi fossero equidistanti dal sender, viene avvantaggiando il veicolo che possiede l'identificato più basso.

Inoltre è stato stabilito un valore soglia per verificare il verso di percorrenza delle vetture, solo nel caso in cui i veicoli si muovono nello stesso verso vengono raggiunti dal messaggio di allarme.

Propagazione dei messaggi di allarme

Dal punto di vista progettuale si utilizzerà ODAM per propagare i messaggi di allarme. Nel momento in cui accade un incidente, si dovrebbero trasmettere messaggi di allarme per informare gli altri veicoli.

Un modo molto semplice potrebbe essere quello di collegare il componente client con l'airbag delle vetture. Nel momento in cui il dispositivo di sicurezza viene attivato, il client inizia la comunicazione con gli altri veicoli.

Oltre l'airbag si potrebbero collegare anche altri dispositivi. Si pensi alle tradizionali 4 frecce che in molti contesti vengono inserite dall'automobilista per avvertire di un improvviso rallentamento a causa di incidenti, lavori stradali o situazioni pericolose.



Figura 3.5: Attivazione dei messaggi di allarme

3.4 802.11p

Nel settembre del 2003 un gruppo di studio per il Wireless Access in Vehicular Environment (denominato WAVE) si riunì per la prima volta. Successivamente nel settembre 2004 fu approvata la richiesta di autorizzazione diventando definitivamente un task group.

WAVE definisce le variazioni al protocollo 802.11 richieste per supportare applicazioni di tipo Intelligent Transport System (ITS). Nel febbraio del 2006 venne completato il primo draft iniziale 1.0 dell'802.11p, esso permette: la comunicazione V2V (Vehicle-to-Vehicle Communication), quella V2I (Vehicle-to-Infrastructure Communication), ma anche tra i veicoli e la strada (applicazioni Road-to-Vehicle).

L'obiettivo del progetto è quello di modificare i protocolli Ieee 802.11 per supportare comunicazioni tra veicoli in movimento a velocità di punta di 200 Km/h, con un raggio di comunicazione fino a 1.000 metri. Le attuali comu-

nicazioni Wi-Fi, che fanno parte della stessa famiglia, perdono di efficacia appena la velocità relativa tra i terminali supera i 10 Km/h.

3.4.1 PHY Layer

L'802.11p [LS06] utilizza un tipo di modulazione multi portante denominata Orthogonal Frequency-Division Multiplexing (OFDM), in essa è presente un numero elevato di sotto portanti, ortogonali tra di loro. Ognuna delle sottoportanti è organizzata attraverso una modulazione di tipo convenzionale (ad esempio una modulazione di ampiezza in quadratura) con un basso symbol rate, mantenendo un data rate simile agli schemi a singola portante. Gli algoritmi OFDM sono generati usando la trasformata di Fourier veloce.

Nella seguente figura sono riportati due tipi di modulazione, nel primo caso si parla di FDM (Frequency Division Multiplexing). Nel secondo caso si parla di OFDM (Orthogonally Frequency Division Multiplexing).

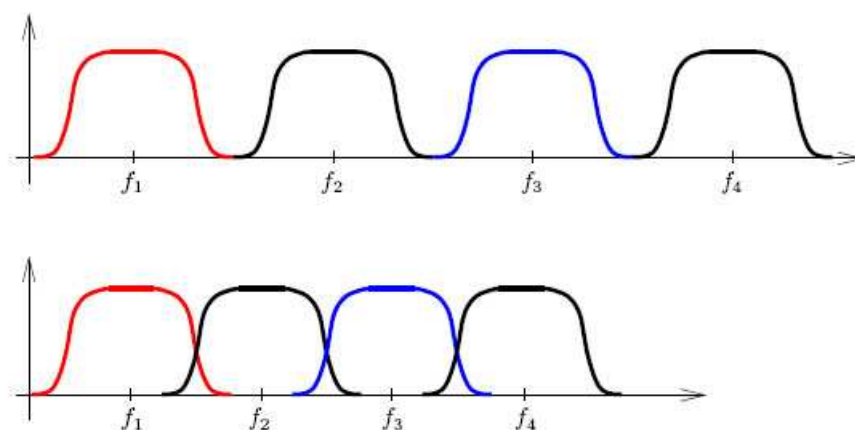


Figura 3.6: Modulazione FDM/OFDM

Un segnale OFDM consiste di N sotto portanti equi spaziate in frequenza

di una quantità f . Il vantaggio primario dell'OFDM rispetto agli schemi a singola portante è l'abilità di comunicare anche in condizioni pessime del canale [Pig04].

Il fading è quel tipo di fenomeno in base al quale un segnale percorrendo distanze più o meno lunghe in condizioni variabili, giunge al ricevitore con un'intensità discontinua. In un contesto come quello VANET il fading è spesso presente, sia perché i nodi si muovono velocemente, sia perché possono variare frequentemente le condizioni di temperatura ed umidità presente nell'atmosfera, in cui il segnale si propaga. Proprio per combattere il fenomeno del fading le informazioni sono codificate prima di essere modulate nel sub-carrier.

Il livello PHY dell'802.11p è simile a quello specificato per la prima versione del 802.11a a differenza per i seguenti punti:

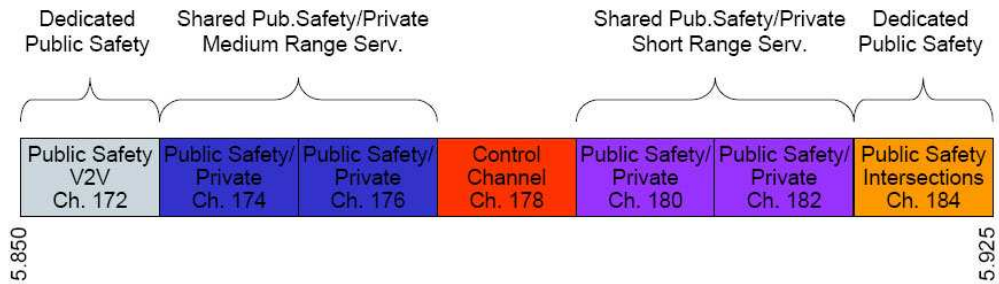


Figura 3.7: 802.11p Livello Fisico

- La banda utilizzata per IEEE 802.11p è 5.9 Ghz . I 75 Mhz sono divisi da sette canali di 10 Mhz ognuno. Il canale centrale è quello di controllo su cui tutti i messaggi di importanza vitale sono mandati in broadcast. I rimanenti canali sono usati per messaggi di servizio, o comunque per informazioni meno importanti, su questi canali è condotta quindi una

comunicazione meno rilevante solo dopo aver negoziato con il canale centrale il controllo del mezzo trasmissivo.

- Per poter supportare un raggio di comunicazione più ampio possibile, sono stati autorizzati valori di EIRP fino ad un massimo di 44.8 dBm (30W). L'EIRP (Effective-Isotropic-Radiated-Power) indica la potenza apparentemente trasmessa verso il ricevitore, assunto che il segnale sia irradiato uniformemente in tutte le direzioni, come un'onda sferica che emana da un punto sorgente; in altre parole, è il prodotto della potenza fornita ad un'antenna e il suo guadagno. $EIRP = G * P = 10(g/10) * P$ [W].
- Per aumentare la tolleranza dell'effetto di propagazione del segnale multi-path, è stato deciso di utilizzare una larghezza di banda di 10 Mhz (10 MHz frequency bandwidth). Tutti i parametri del time domain sono inoltrati due volte, così facendo da un lato vengono attenuati i problemi relativi all'effetto Doppler dall'altro si ha un'attenuazione delle interferenze causate dal multipath.
- Per contro il data rate è dimezzato.

L'effetto Doppler è un cambiamento apparente della frequenza o della lunghezza d'onda di un'onda percepita da un osservatore che si trova in movimento rispetto alla sorgente delle onde. Per quelle onde che si trasmettono in un mezzo, come le onde sonore, la velocità dell'osservatore e dell'emettitore vanno considerate in relazione a quella del mezzo in cui sono trasmesse le onde. L'effetto Doppler totale può quindi derivare dal moto di entrambi, ed ognuno di essi è analizzato separatamente.

In questo contesto il range massimo di trasmissione è di 1000 metri.

3.4.2 MAC Layer

802.11p utilizza un meccanismo denominato ECDA, previsto anche per l'802.11e [HK05]. Quest'ultimo è stato approvato nel 2005 ed apporta significativi miglioramenti del Quality of Service (QoS)[OMWE07] nell'ambito delle reti Wi-Fi. Lo standard è particolarmente indicato per tutte quelle applicazioni che sono delay-sensitive, quali possono essere il Voice over Wireless IP e lo Streaming Multimedia.

ECDA prevede l'ascolto del canale prima di poter comunicare ed un periodo di back-off random. Il tempo di attesa prefissato è determinato dal parametro AIFSN e dalla Contention Window (CW). Il parametro iniziale di CW è posto pari a CWmin ed ogni volta che la comunicazione fallisce, il valore di CW raddoppia fintanto che non si sia raggiunta il valore di CWmax.

La priorità dei pacchetti viene effettuata assegnando un valore (AC) ad ogni singolo pacchetto.

Con EDCA, il traffico ad alta priorità ha una più alta probabilità di essere spedito che non il traffico a bassa priorità: una stazione ad alta priorità aspetta un po' meno in media che una stazione a bassa priorità.

In aggiunta ogni livello di priorità ha una Transmit Opportunity (TXOP) assegnata. Una TXOP è un intervallo di tempo limitato durante il quale una stazione può inviare più frame possibile (fino a che la durata della trasmissione non si estende oltre la massima durata del TXOP).

In uno scenario ad alta mobilità come quello delle reti VANET, l'intervallo di tempo durante il quale i veicoli sono in comunicazione tra di loro è molto limitato. Per ottimizzare questi tempi l'overhead di informazioni contenute in un pacchetto deve essere necessariamente basso. Proprio per questo nessun frame è scambiato prima della trasmissione dati.

In un contesto WAVE il basic service set (BSS) è annunciato da una

STA che trasmette ad intervalli regolari il WAVE service announcement frame (WSA). Quest'ultimo è simile al beacon frame dello standard 802.11 tradizionale.

Non è necessario scambiare nessun frame né di autenticazione né di associazione, affinché un terminale possa associarsi ad una WBSS.

Visto che il beacon frame non è utilizzato, non è possibile usufruire della funzione TSF (timing synchronization function) per la sincronizzazione dei dispositivi dello stesso BSS. Per assolvere questa funzione viene utilizzato un dispositivo esterno, quale potrebbe essere un ricevitore GPS che fornisce una stima dell'ora UTC molto accurata.

3.4.3 Conclusioni

Come precedentemente illustrato le reti VANET sono particolari reti MANET i cui nodi sono veicoli che si muovono molto velocemente, i nodi hanno la necessità di comunicare in maniera rapida e affidabile.

L'utilizzo dell'802.11p apporta notevoli vantaggi in contesti VANET, come abbiamo visto infatti, a discapito del data rate, aumenta il raggio di comunicazione così come la tolleranza. Viene effettuata un gestione del QoS migliore rispetto al 802.11 b/g, inoltre l'assenza di alcune fasi come quella di associazione e sincronizzazione rende la comunicazione molto più veloce.

Capitolo 4

Il progetto

La fase di progetto effettuata è stata finalizzata a memorizzare lato server le informazioni ricevute da un componente client già esistente. La funzione del server è quella non solo di mantenere una sorta di storico dei dati ma anche di elaborare le informazioni permettendo di aggiungere funzionalità aggiuntive ai tradizionali servizi presenti nei ricevitori GPS.

Un occhio di particolare riguardo è stato dato anche al potenziale aspetto economico del progetto. Sostanzialmente l'intento era quello di sfruttare infrastrutture gratuite o comunque già esistenti, per fornire servizi ad un ente comunale e quindi ai cittadini al minor costo possibile con un investimento minimo.

In questo capitolo verrà illustrata l'architettura utilizzata, le soluzioni studiate e le possibili alternative.

4.1 La piattaforma LAMP

Si è deciso di utilizzare un'architettura di tipo LAMP. Quest'ultimo è un acronimo che prende il nome dalle lettere iniziali dei componenti software che costituiscono la piattaforma, attualmente di gran lunga la più utilizzata nel web per lo sviluppo di applicazioni web dinamiche [Wik07a].

I programmi che la costituiscono sono:

- Sistema Operativo: Linux
- Web Server: Apache
- Database Relazione: MySQL
- Linguaggio di scripting: PHP.

Linux, Apache MySQL e PHP sono alla base di molte applicazioni web, si pensi per esempio a molti siti di e-commerce oppure applicazioni come WordPress che devono supportare un elevato volume di traffico. Solitamente l'architettura LAMP è presente di base in molte delle distribuzioni linux, ed ormai è integrata talmente bene che non si ha la percezione che i componenti siano in ascolto sulla macchina.

A causa della natura particolare dell'applicazione si è deciso di configurare al meglio i vari componenti in base alle esigenze specifiche delle reti vanet e degli scenari che si potranno delineare nei prossimi anni con un aumento esponenziale sia dei dispositivi GPS che dell'accesso ad Internet attraverso dispositivi Wireless.

4.1.1 Come funziona una tipica applicazione LAMP?

Un'applicazione LAMP è scritta in PHP, linguaggio che gira come parte del web server Apache a sua volta ospitato su un server Linux.

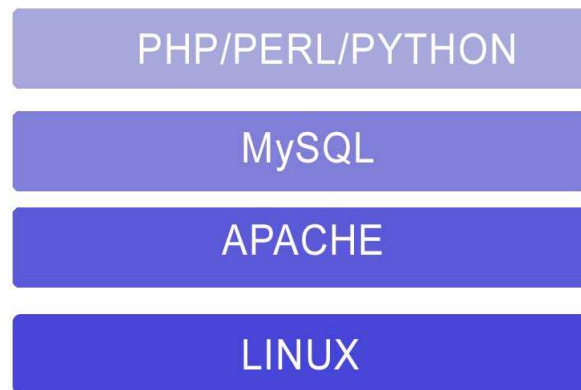


Figura 4.1: Tipica architettura LAMP

L'applicazione PHP prende informazioni dal client attraverso le richieste URL, solitamente da campi form attraverso GET o POST. Se è necessario il componente server riceve informazioni da un database MYSQL (che potrebbe girare sulla stessa macchina LINUX), combina le informazioni ricevute con diversi tag per produrre un documento HTML o XML che successivamente restituisce al client. Il processo può verificarsi in parallelo nel momento in cui molti utenti accedono al sistema.

Oltre gli elementi dinamici sono richiesti anche gli elementi statici come immagini, javascript e CSS che permettono una formattazione più gradevole del documento restituito in output.

In un contesto LAMP è il database che fornisce gli elementi dinamici, non a caso il client potrebbe notare un certo ritardo causato dal tempo necessario per effettuare la query. Il web server deve essere abile ad eseguire gli script velocemente ma anche a gestire molte richieste concorrenti.

4.2 Ottimizzazione componenti LAMP

Dato il particolare scenario che si presenta in un contesto come quello delle reti veicolari, ho preferito ottimizzare l'architettura per tentare di ottenere un incremento delle prestazioni. Visto che il progetto si trova ancora in una fase prototipale non è stato possibile effettuare veri e propri benchmark, si è riusciti però a trarre delle deduzioni su come possono essere impostate le configurazioni dei vari componenti.

Un modo molto empirico per misurare le performance dell'architettura è osservare l'uso del CPU (Central Processing Unit) del server, ed osservare se e quando risulta elevato. Altro parametro da tenere in considerazione è l'utilizzo di banda effettuato in passato per intuire quelle che sono le prospettive future e gli upgrade necessari nel sistema.

Altra strada è quella di ottimizzare le configurazioni dei vari componenti, osservando se ogni modifica apportata comporta un miglioramento o un peggioramento delle prestazioni complessive. Quindi in fase di test viene fortemente raccomandato apportare una modifica per volta.

4.2.1 Ottimizzazione LINUX

Si è scelto un sistema operativo open source sia per esigenze economiche (esso infatti è totalmente gratuito) sia perché un web server che gira sotto UNIX è meno vulnerabile rispetto IIS di Microsoft o altri. Inoltre dal punto di vista delle risorse hardware i server linux ne richiedono un po' meno rispetto un Server Windows, quindi la macchina che si utilizzerà dovrebbe essere meno costosa. È dimostrato che i server unix hanno un tempo di obsolescenza decisamente maggiore, questo permetterebbe il riutilizzo anche di hardware già disponibile. Il sorgente è aperto ed esistono comunità di milio-

ni di sviluppatori in tutto il mondo, per cui il supporto è gratuito e molto rapido.

Esistono molti modi per misurare il tempo di risposta di un web server, un modo molto intuitivo e disponibile in modo nativo in molte distribuzioni Linux (tra cui Ubuntu) è il comando:

```
$ curl -o /dev/null -s -w %time_connect:%time_starttransfer:%time_total  
http://www.google.it
```

I parametri specificati permettono di:

- -o ri-indirizzare l'output video su una particolare periferica
- -s Silent mode. Non mostra i messaggi di errore
- -w definisce quali parametri mostrare dopo che l'operazione è stata completata con successo.

Nel nostro caso sono richiesti:

time-connect: Il tempo impiegato per stabilire la connessione TCP al server;

time-starttransfer: Il tempo impiegato dal web server per inoltrare il primo byte di dati dopo che la richiesta è stata effettuata

time-total: Il tempo totale impiegato per completare la richiesta;

Il tempo indicato è misurato dall'inizio della transazione, prima del DNS lookup. Quindi se il risultato di output fosse il seguente:

0.081:0.272:0.779 potremmo concludere che sono trascorsi $0.272 - 0.081 = 0.191$ secondi affinché il web server processi la richiesta e inoltri il primo byte, inoltre il client ha impiegato $0.779 - 0.272 = 0.507$ secondi per scaricare i dati dal server.

Il comando di curl risulta molto interessante, ma visualizza il tempo di risposta di una sola risorsa. Molto spesso l'esigenza è quella di monitorare

un'applicazione web che è composta da molteplici elementi, risultano utili in questo caso estensioni per firefox come Tampare Data o FireBug, che monitorizzano il tempo di risposta di tutte le richieste effettuate dal browser.

In Tampare Data sono mostrati diversi elementi quali il metodo della richiesta, lo status code della risorsa, e i tempi di caricamento, in questo modo si ha un quadro complessivo dell'andamento dell'applicazione web. È possibile visualizzare un grafico con i tempi di caricamento di tutte le risorse, in grado di fornire un riscontro immediato sull'andamento del web server.

Prima di ottimizzare Apache, PHP e MySQL è necessario assicurarsi che i componenti basilari del kernel linux stiano funzionando correttamente, ed è logicamente necessario eliminare tutti i servizi che non risultano necessari.

Tcp/Ip

È possibile modificare alcune opzioni dell'implementazione del TCP/IP per allocare maggiore memoria migliorando le performance dell'intera architettura [WR06].

In Linux i parametri del kernel sono settati attraverso l'interfaccia *proc*, fortunatamente in ubuntu è disponibile il programma *sysctl* che permette una gestione più agevole attraverso il file di configurazione */etc/sysctl.conf*. Nel listato seguente vengono mostrati alcuni parametri che permettono di incrementare notevolmente le prestazioni.

```
#Use TCP syncookies when needed
net.ipv4.tcp_syncookies = 1
#Enable TCP window scaling
net.ipv4.tcp_window_scaling = 1
#Increase TCP max buffer size
```

```
net.core.rmem_max = 16777216
net.core.wmem_max = 16777216
#Increase Linux autotuning TCP buffer limits
net.ipv4.tcp_rmem = 4096 87380 16777216
net.ipv4.tcp_wmem = 4096 65536 16777216
#Increase number of ports available
net.ipv4.ip_local_port_range = 1024 65000
```

net.ipv4.tcp_syncookies=1 Abilita il sys cookie, questa tecnica evita che il server faccia cadere le connessioni nel momento in cui la coda di SYN risulta piena. In modo molto semplice il server continua a mandare al client un appropriato SYN+ACK senza però inserirlo nella coda di SYN.

Solo successivamente se il server riceve un ulteriore ACK di conferma dal client, esso ricostruisce la coda di ingresso di SYS utilizzando le informazioni contenute nel pacchetto TCP ricevuto.

Il syn cookie risulta compatibile con tutte le implementazioni del TCP e permette di evitare attacchi conosciuti come SYN flood. Quest'ultimo è un particolare tipo di attacco DOS, che consiste nell'inoltrare molte richieste SYN al server, il client malefico pur ricevendo dal server i SYN-ACK volontariamente non inoltra l'ACK di conferma, in questo modo tutte le connessioni sono in uno stato denominato half-opened consumando molte risorse lato server. Nel momento in cui un qualsiasi altro utente fa una richiesta al web server, quest'ultimo rifiuta la connessione.

```
net.ipv4.tcp_window_scaling = 1
```

Permette ai client di scaricare i dati ad una velocità maggiore. L'opzione window scaling permette di aumentare il valore massimo della congestion windows definito essere nella RFC 1323 di 65,536 bytes. La congestion windows potrebbe essere aumentata fino ad un valore massimo di 1 gibabyte

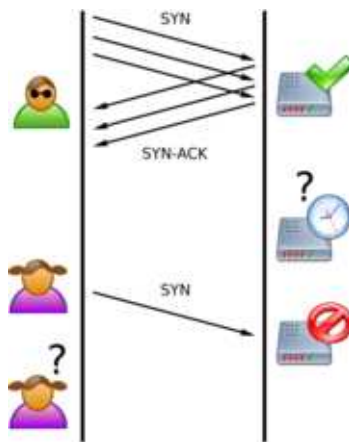


Figura 4.2: Esempio di attacco SYN flood

(1,073,741,824 bytes). Sostanzialmente in questo modo viene aumentato il numero di pacchetti che può essere inoltrato senza avere un acknowledgment, in un contesto come quello delle reti veicolari, tale scelta risulta molto indicata perché il client deve mandare informazioni nel minor tempo possibile.

Le quattro successive configurazioni incrementano il buffer sia in entrata che in uscita. In questo modo l'applicazione può liberarsi dei suoi dati in minor tempo riuscendo successivamente a servire un'altra richiesta.

```
net.ipv4.ip_local_port_range = 1024 65000
```

Questa configurazione aumenta il numero di porte locali disponibili per l'uso, così facendo aumenta il numero massimo di connessioni che possono essere servite nello stesso tempo.

Configurazione del disco

Il sottosistema disco ha un'importanza vitale in un'architettura LAMP. Solitamente il sistema operativo registra informazioni sia sulla data di creazione di un file sia sulla data in cui è stata effettuata l'ultima modifica. Attraverso

il campo `atime` si tiene traccia dell'ultimo accesso al file, è il file system che sostituisce il record precedentemente con il timestamp relativo all'ultimo accesso [IBM07]. Dal momento in cui nell'applicazione, come vedi vedrà successivamente, il timestamp sarà un dato rilevante solo nella query di immissione al database ma non interessa tener traccia dell'ultima modifica dei file nel file system, quindi è possibile non indicare tale opzione nel file `fstab`. L'`fstab` indica i dispositivi che devono essere montati con le rispettive opzioni, nel nostro caso l'opzione da specificare è `noatime`. Così facendo aumenta la velocità di accesso ai dati. Dopo aver effettuato le modifiche è necessario un remount del file system.

Per avere una stima della velocità di trasferimento del disco è possibile usare il comando `hdparm`:

```
# hdparm -t /dev/hda
>Timing buffered disk reads: 182 MB in 3.02 seconds =
60.31 MB/sec
```

L'esempio precedente indica che il disco riesce a leggere i dati ad una velocità di circa 60 MB/s.

4.2.2 Ottimizzazione Apache

APACHE è presente di default in quasi tutte le distribuzioni, è un applicazione Open Source, multi piattaforma, sviluppata secondo le specifiche e gli standard più aggiornati dell'HTTP.

Dopo aver toccato picchi del 69% nel 2005, attualmente i server apache rappresentano più del 50% del mercato dei web server (fonte <http://news.netcraft.com/>).

Apache è un web server altamente configurabile, esso è costituito da molti

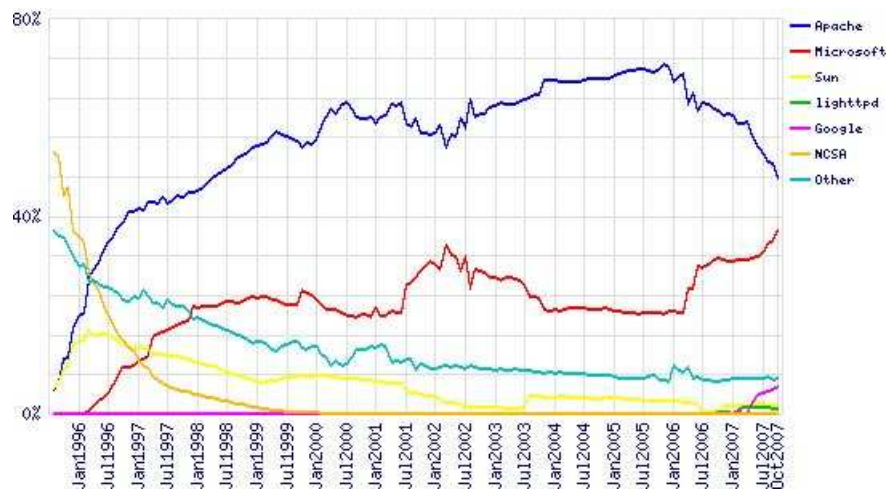


Figura 4.3: Tipologia di Web Server Utilizzati

moduli ognuno dei quali comporta un prezzo in termini di tempo di esecuzione dell'intera architettura. Come precedentemente indicato si consiglia sempre di utilizzare solo i moduli necessari per l'applicazione.

MPMs è un modulo che permette di gestire le connessioni multiple e fare il dispatching delle richieste. Sono disponibili diversi i seguenti pacchetti MPM:

- apache2-mpm-prefork
- apache2-mpm-worker
- apache2-mpm-perchild - Sviluppo non completo
- apache2-mpm-itk (solo sulla Etch) - Sperimentale
- apache2-mpm-event (solo sulla Etch) - Sperimentale
- apache2-mpm-threadpool (solo sulla Sarge) - Sperimentale

Prendendo in considerazione solo quelli standard, la scelta è ricaduta sul tradizionale modulo prefork perché sebbene il worker utilizza processi multipli, ognuno dei quali con tread multipli, garantendo ottime performance con

un basso overhead. Attualmente esistono alcuni problemi di stabilità specialmente se si utilizzano librerie come quella del php4 che non sono compilate in maniera thread-safe.

Il modulo prefork può essere configurato attraverso il file `apache2.conf`:

```
StartServers 50
MinSpareServers 15
MaxSpareServers 30
MaxClients 300
MaxRequestsPerChild 4000.
```

Nel modello prefork, per ogni richiesta viene creato un nuovo processo. Il parametro `StartServers` indica quanti processi server figlio verranno creati all'avvio di apache.

Il 2° e 3° valore indicano il numero min e max di processi server figli in attesa di richieste. Se esiste un altro server sulla stessa macchina fisica (per esempio MySQL) i valori di spare server devono essere diminuiti.

Il valore `MaxClients` indica il numero massimo di processi attivi consentito. L'obiettivo è quello allocare i processi o threads senza causare un eccessivo swap del server. Se sono effettuate più richieste rispetto a quelle indicate, tutte quelle in eccesso verranno bloccate. Sarà necessario configurare attentamente questo parametro in funzione sia del traffico che della percentuale di veicoli che utilizzano il nostro dispositivo. Ogni processo può effettuare molte richieste consecutive, esso però verrà terminato (ed eventualmente riavviato) al raggiungimento del numero di richieste specificato in `MaxRequestsPerChild`. In questo modo la memoria rimasta allocata dagli script verrà liberata.

Ogni richiesta che il server Apache processa passa attraverso un complicato set di regole che dettano condizioni o speciali istruzioni che il web server deve seguire. Di seguito si elenca la configurazione di particolare direttive:

```
.....  
<Directory >  
    Options FollowSymLinks  
</Directory >  
.....
```

Questa direttiva permette di seguire i link simbolici, in questo modo si ottiene un incremento delle prestazioni. Infatti nel caso in cui questa opzione fosse disabilitata il web server dovrebbe ogni volta controllare ogni componente del file system ad assicurarsi di non accedere a link simbolici, questo comporta un overhead per quanto concerne l'attività del disco. È logico che in questo modo si potrebbe incorrere in problemi di sicurezza per l'accesso non autorizzato su file del file system. In ogni caso si deve sempre scegliere tra funzionalità e rischi.

In apache ogni directory può contenere un file denominato `.htaccess`. Quest'ultimo contiene direttive aggiuntive che il web server processa ad ogni richiesta. Solitamente viene utilizzato per effettuare il controllo degli accessi e per l'URL rewrite. Visto che nel progetto non ci sono necessità di questo tipo, si è scelto di non utilizzare il file `.htaccess` inserendo tutte le direttive necessarie direttamente nel file di configurazione globale `http.conf`. Così facendo viene diminuita l'attività del disco fisso per ogni richiesta.

HostnameLookups off

Effettua il reverse lookup su ogni indirizzo ip. Se l'impostazione è posta su `off` viene mantenuta traccia solo dell'indirizzo ip e non del nome del dominio, in questo modo però si miglioreranno sia i tempi di risposta del web server sia l'utilizzo della banda. In un'applicazione per reti VANET dove tra l'altro è necessario garantire l'anonimato delle informazioni, inoltrare al server tale impostazione risulta non necessaria (quindi è consigliabile metterla nello

stato di off).

KeepAlive On

L'opzione KeepAlive, se settata On, permette di utilizzare, come da specifiche HTTP/1.1, la stessa connessione TCP per inviare più file. Per migliori performance è consigliato settarla On evitando che venga aperta una connessione TCP per ogni richiesta HTTP.

MaxKeepAliveRequests 100

Specifica il numero di richieste permesse per connessione (sempre che l'opzione KeepAlive venga settata On). Se questa opzione è settata a 0, una connessione rimane attiva e file continuano ad essere scambiati fino a quando non va in timeout.

KeepAliveTimeout 2

Specifica quanto tempo, in secondi, Apache deve attendere per una successiva richiesta prima di chiudere la connessione. Un valore di 15 secondi risulta molto buono (parlando sempre di performance di Apache). Un basso valore di timeout (nel nostro caso 2) assicura che i processi non stiano troppo a lungo nella fase idle aspettando una richiesta che potrebbe non arrivare.

Visto che le informazioni sia ricevute da server sia inviate in risposta sono file di testo in xml, si potrebbe pensare di utilizzare un modulo di apache che permette la compressione dei dati. Per maggiori informazioni sul mod deflate si rimanda al sito http://httpd.apache.org/docs/2.0/mod/mod_deflate.html

4.2.3 Ottimizzazione PHP

Php è un linguaggio di scripting open source. Nato nel 1994 ad opera del danese Rasmus Lerdorf, inizialmente fu sviluppato per la generazione di pagine web dinamiche.

Nel momento in cui uno script php viene richiesto, il motore PHP legge

lo script e lo compila in quello che viene definito Zend opcode, solitamente il codice opcode è eseguito dal motore engine del PHP e successivamente gettato via.

L'utilizzo della cache permetterebbe di riutilizzare il codice opcode in una successiva richiesta dello stesso script. La cache quindi influirebbe in modo positivo sul tempo di esecuzione della richiesta.

Esistono molti opcode caches disponibili sul mercato : APC, eAccelerator, Zend Platform ed il giovane Xcache.

La scelta è ricaduta sul eAccelerator, distribuito con licenza open source, perché offre anche altre funzionalità, come l'ottimizzazione del codice oppure la possibilità di codificare i propri script in maniera da poterli distribuire senza sorgenti. Inoltre alcuni test effettuati [Ipe] mostrano come eAccelerator risulti di gran lunga superiore agli altri prodotti nel momento in cui si modifichi il codice e vi sia la necessità di rigenerare la cache.

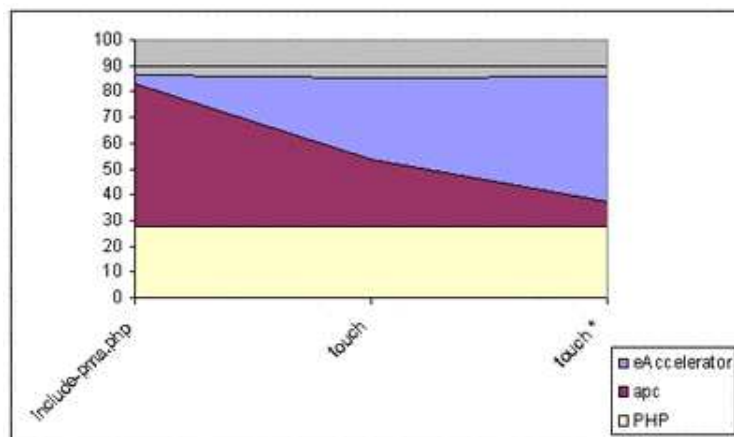


Figura 4.4: Confronto php cache

Nel file *accelerator.ini* è possibile specificare la dimensione della memoria cache condivisa, il valore da specificare è in megabyte. Ritengo che un valore di 64 Mbyte sia sufficiente perché gli script che saranno richiesti al web server

saranno grossomodo sempre gli stessi, quindi non sarà necessario una grande porzione di spazio disco.

```
eaccelerator.shm_ttl = 60
```

Dopo che si è ecceduto il valore di cache, tutti gli script che non sono stati utilizzati nel lasso di tempo specificato nel ttl time vengono cancellati dalla cache. Un valore di 60 secondi è più che ragionevole anche se l'intento dell'applicazione è quello di mantenere tutte le funzioni php nella cache del motore engine di PHP.

In funzione dell'applicazione in oggetto possono essere modificati alcuni valori di default presenti nel file php.ini.

```
max_execution_time =120
```

Visto che vi sono molte operazioni sia di I/O che di accesso al database, si ritiene necessario aumentare il valore di default.

```
max_input_time=120
```

Indica il valore in secondi che l'applicazione aspetta per ricevere i file dal client. Visto che vi potrebbero essere file xml abbastanza consistenti o problemi di congestione della rete internet (che viene utilizzata per inviare i dati) o altri problemi legati ad uno contesto atipico, il valore raccomandato di questo parametro è stato raddoppiato portandolo a 120 secondi.

```
memory_limit = 50 MB
```

È il limite massimo di memoria che uno script può consumare, un valore di 50mb soddisfa ampiamente le esigenze dell'applicazione.

```
output_buffering = 4096
```

Questa impostazione fa sì che il motore php raccolga tutto l'output in un buffer prima di inviarlo al computer del client; questa tecnica ha il vantaggio di attendere tutte le chiamate delle funzioni.

4.2.4 Ottimizzazione MySQL

Per migliorare le prestazioni del server Mysql solitamente sono tre le strade che si possono intraprendere:

- Ottimizzare le query.
- Ottimizzare la configurazione del server MySQL.
- Migliorare l'hardware sottostante, in modo particolare il sottosistema disco.

Per quanto riguarda l'hardware avere un sistema multi-processore solitamente fa aumentare in maniera considerevole le prestazioni (solitamente di un fattore che va da 4 a 8). Inoltre la velocità di accesso del disco e la configurazione in raid degli stessi sono fattori da tenere in considerazione.

In un server MySQL le tabelle di dati sono memorizzate sul file system. Gli index sono un mezzo utilizzato dal server per trovare una particolare riga all'interno della tabella senza la necessità di scorrerla tutta [YZ07]. Quando si ha la necessità di scorrere tutta la tabella, questa operazione viene denominata table scan. Molto spesso però si è interessati solo ad un piccolo subset di dati, ed un'operazione di table scan fa compiere molte operazioni di I/O consumando molto tempo.

Un uso insufficiente degli index associato a query mal strutturate fanno insorgere problemi che sono direttamente proporzionali alla grandezza della tabella. In uno scenario come quello in discussione si presume che le tabelle crescano enormemente in particolari orari della giornata (prima mattina, orario di chiusura degli uffici nel pomeriggio).

È possibile configurare attraverso il file `my.cnf` il server mysql affinché memorizzi in un log separato le slow query. Quest'ultime sono le query che eccedono un determinato intervallo di tempo.

```
[mysqld]
# enable the slow query log , default 10 seconds
log_slow_queries = /nuovo/percorso/
# log queries taking longer than 5 seconds
long_query_time = 5
# log queries that don't use indexes even if
# they take less than long_query_time
# MySQL 4.1 and newer only
log-queries-not-using-indexes
```

Attraverso questi parametri si loggano tutte le query che impiegano più di cinque secondi e tutte le query che non utilizzano gli index. L'utilizzo dei file di log risulta molto utile in fase di debug.

Ogni volta che una query è effettuata, il database compie sempre lo stesso lavoro: analizza la query, determina in che modo eseguirla, carica le informazioni dal disco restituendo il valore dell'interrogazione al client. Il server MySQL ha una funzione denominata *query cache* che immagazzina in memoria i risultati della query effettuate precedentemente, in molti contesti incrementa le performance drasticamente. Solitamente la query cache è disabilitata di default. È possibile abilitarla aggiungendo la seguente stringa nel file di configurazione *etc/my.conf*:

```
query_cache_size = 32M
```

È possibile anche monitorare le query effettuate attraverso la cache con il seguente comando :

```
SHOW STATUS LIKE 'qcache\%';
```

set-variable=max_connections=500

L'idea alla base del parametro `max_connections` è la stessa di `MaxClient` per Apache, solamente un numero di connessioni viene servito ed autorizzato.

set-variable=wait_timeout=10

Termina le connessioni che non compiono operazione nell'intervallo di tempo specificato.

max_connect_errors = 50

Questo parametro è utilizzato per risolvere problemi di sicurezza [MEZ03]. Se un host tenta di connettersi al server più volte, non avendo le autorizzazioni per farlo, al 50° tentativo viene inserito nella lista degli host da bloccare.

4.3 Componente CLIENT

Il CLIENT deve essere installato su un palmare o su un dispositivo GPS, che riesca ad eseguire un'applicazione java ed inviare le informazioni attraverso la rete internet sfruttando un'infrastruttura già esistente e completamente gratuita. Una prima sperimentazione potrebbe partire sfruttando la rete Iperbole, che copre molte zone del centro cittadino di Bologna. Successivamente una stazione-base WiMAX basterebbe per coprire un'area per un raggio di 50 KM, sufficientemente grande per esigenze comunali.

Il progetto nasce da un componente client già precedentemente sviluppato. Il lavoro iniziale è stato quello di analizzare il file KML prodotto dal client individuandone possibili miglioramenti.

In un secondo momento è stato realizzato un componente server che prende in input i file kml prodotti, ed attraverso un parser XML ed un'iterazione tramite le API di Via Michelin, memorizza in un database i dati relativi alla velocità delle strade percorse.

Nel file KML sostanzialmente vengono salvate le coordinate GPS di latitudine e longitudine ad intervalli regolari.

Un estratto del file è il seguente :

```
<coordinates>
11.35613587108126086,44.49679205201381885,0,
11.35626461815762933,44.49677059416775744,0,
11.35632899169581357,44.49657747355320472,0,
11.35613587108126086,44.49629852155440636,0,
11.35538484646911143,44.4965345578610819,0,
11.35454799047271634,44.49679205201381885,0,
11.35368967663025985,44.49702808832049438,0,
```

```
11.35353947170782996,44.49741432954959981,0,  
</coordinates>
```

KML è un linguaggio basato su XML che serve per gestire coordinate geografiche di latitudine, longitudine ed altezza, ed è utilizzato soprattutto nelle applicazioni proprietarie di Google come i Google Earth, Google Maps e Google Mobile [Goo].

Nella figura seguente si riporta la visualizzazione iniziale del file kml prodotto dal CLIENT:

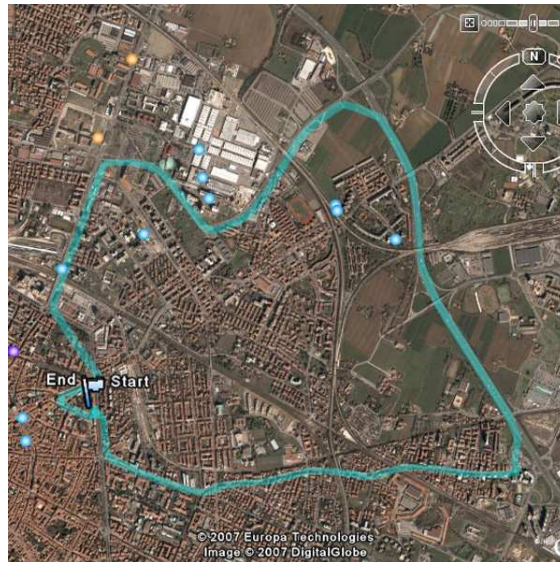


Figura 4.5: Visualizzazione del KML iniziale

Sostanzialmente è possibile visualizzare il percorso del singolo veicolo attraverso la concatenazione dei punti individuati dalle coordinate GPS presenti nel file KML.

Analizzando attentamente il percorso si notano zone in cui il ricevitore GPS si è spento accidentalmente, in questi casi viene visualizzata una linea continua che attraversa alcuni edifici, non risultando quindi fedele alle strade esistenti ed effettivamente percorse.

Un parser XML è un programma che permette di analizzare e manipolare un file XML o KML, dopo che il file viene caricato è possibile modificarlo attraverso il DOM o SAX .

Il DOM (Document Object Model) è un linguaggio ad oggetti indipendente della piattaforma utilizzato per rappresentare HTML, XML e linguaggi simili quale può essere appunto il KML.

La scelta di utilizzare il DOM è stata dettata dal fatto che tale linguaggio supporta la navigazione del documento in qualsiasi direzione (per esempio dal padre al fratello successivo) permettendo modificazioni arbitrarie. Probabilmente il DOM è la migliore suite per tutte quelle applicazioni in cui il documento deve essere acceduto spesso o i cui nodi devono essere riscritti o riposizionati. Nel caso invece di applicazione strettamente sequenziale il modello SAX risulta essere il più indicato perché più veloce nonostante faccia un uso minore di risorse.

Visto che non tutti i browser presentano lo stesso supporto per i vari livelli del DOM si è deciso di utilizzare un browser basato su un motore di rendering Gecko, scritto in C++ e progettato per supportare standard aperti. Gecko è multipiattaforma ed è supportato da molti sistemi operativi (windows, unix, mac os etc ...).

La scelta è ricaduta su Mozilla Firefox, browser basato su Gecko che supporta totalmente le specifiche DOM level 1, in maniera parziale le versioni level 2 e 3. Inoltre questo browser permette la comunicazione asincrona (AJAX) che sarà utilizzata in una fase successiva del progetto.

Il documento xml fornito dal client viene caricato in maniera integrale, successivamente il parser analizza solo il contenuto del tag kml <coordinates >in cui vengono indicate le coordinate di latitudine e longitudine relative al percorso effettuato.

A questo punto è possibile calcolare la velocità media relativa al tratto di strada percorso. Sostanzialmente si tratta di un problema di geodesia, viene utilizzata la Great-circle distance Formula per determinare la distanza tra 2 punti GPS [Cen].

$$DISTANCE = 69.1 \times (180/\pi) \times \arccos[\sin(LAT_1) \times \sin(LAT_2) + \cos(LAT_1) \times \cos(LAT_2) \times \cos(LONG_2 - LONG_1)] \quad \Lambda \quad (3)$$

Figura 4.6: Calcolo della distanza

Attraverso una funzione viene determinata la distanza esistente tra due punti. Successivamente viene calcolato un coefficiente che è funzione dell'intervallo di tempo esistente tra due rilevazioni successive. Moltiplicando i due valori si riesce ad ottenere la velocità media del tratto in Km/h.

A questo punto è presente un'informazione in più rispetto al file originario del client, ovvero la velocità di percorrenza. Per mantenere questa informazione in un database viene effettuata una query che permette di memorizzare le coordinate iniziali e finali del tratto di strada percorso ed inoltre il valore della velocità media.

In base alle informazioni contenute in database è possibile riscrivere il file KML, colorando il tratto di strada in funzione della velocità di percorrenza.

Utilizzando il primo principio definito da Norman per riassumere il design basato sull'utente, bisogna:

Usare sia la conoscenza presente nel mondo esterno che la conoscenza interiorizzata.

Secondo Norman infatti il programmatore deve sfruttare sia la conoscenza presente nel mondo esterno, perchè così facendo l'utente ha meno cose

riscontro visivo immediato sullo stato di viabilità generale.

4.4 Componente SERVER

Il componente server realizzato prende in un input i file prodotti dal client, analizza il percorso svolto, memorizza le informazioni relative sia al tratto di strada (nome della via percorsa) che la velocità media. In questo modo il programma è in grado di avere una buona conoscenza dello stato di viabilità complessivo quindi può non solo determinare la velocità media reale di una particolare via ma anche indicare possibili percorsi alternativi. In questo paragrafo verranno illustrate le caratteristiche dell'applicazione, il database creato e i componenti esterni con cui il programma dialoga.

4.4.1 Upload dei file KML

Come detto inizialmente si presuppone che il client possa lanciare l'applicazione java ed inviare su una cartella pubblica del web server i file KML creati durante il tragitto.

Compito dell'applicazione è quello di controllare se sono stati ricevuti nuovi file dall'ultimo accesso e procedere al parser degli stessi.

Il parser analizza solo le coordinate presenti nel tag `<coordinates>` ovvero quelle relative al percorso realmente intrapreso.

A questo punto il programma interagisce con un componente esterno il Via Michelin Web Service.

4.4.2 ViaMichelin Web Services

I web services di ViaMichelin mettono a disposizione, sotto forma di componenti di sviluppo, servizi web per la generazione di mappe, calcolo degli itinerari, geocodificazione, verifica di indirizzi e ricerca di prossimità.

I servizi sono forniti in modalità ASP, quindi totalmente indipendenti dal

canale di diffusione (PC, palmare, sistema di navigazione, cellulare). Tramite gli standard tecnologici XML/SOAP/WSDL e si interfacciano facilmente con l' applicazione sviluppata [Via07].

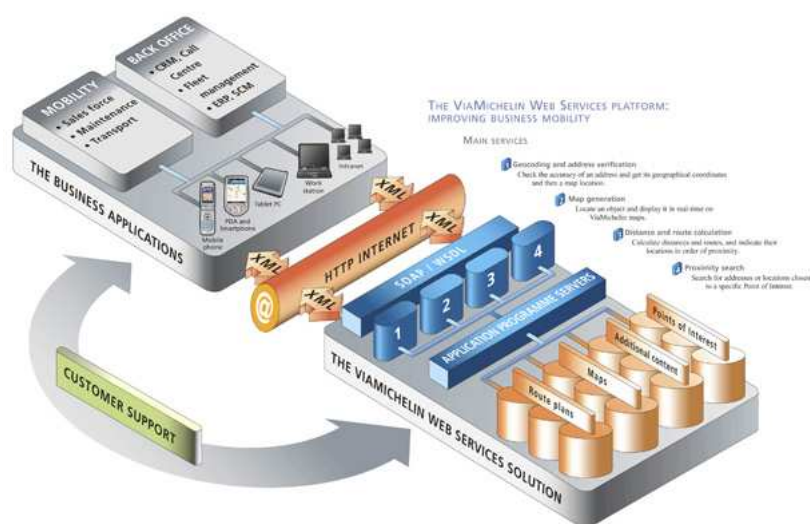


Figura 4.8: ViaMichelin Web Services

I Web Services di ViaMichelin, permettono di accedere ai servizi cartografici grazie ad una soluzione aperta indipendente dal sistema operativo.

Tutti gli altri sistemi, come per esempio quello di Google Maps, solitamente rilasciano la funzione di Geocoding che permette di ottenere le coordinate geografiche x (longitudine) e y (latitudine) immettendo un indirizzo corretto. È possibile localizzare una città, una via o un punto di riferimento (municipio, stazione...) e visualizzarlo su una mappa. Solitamente il geocoding è ottimizzato per riconoscere gli indirizzi di vari paesi e si adatta alle norme di codifica degli indirizzi proprie di ciascuna nazione.

Si è preferito utilizzare le API di Via Michelin, piuttosto che quelle di Google Maps, perché è presente la funzione di ReverseGeocoding. Quest'ultima è una funzione che trasforma le coordinate geografiche di qualsiasi oggetto

fisso o mobile in un indirizzo esatto. Ad esempio, si può conoscere in tempo reale la posizione esatta di un veicolo equipaggiato con un sistema GPS che invia le coordinate all' applicazione.

In ogni caso è bene ricordare che i servizi in ASP di questo tipo sono in continua evoluzione, in un periodo relativamente breve di nove mesi in cui tale documento è stato scritto sono stati apportati miglioramenti significativi, quindi non è escluso che tra poco anche Google fornisca la funzione di ReverseGeocoding.

Nel progetto sviluppato viene utilizzata la funzione di ReverseGeocoding per determinare i nomi delle vie del percorso effettuato. Sono state utilizzate la tipologia di API gratuite, fruibili attraverso Javascript. Quest'ultimo è un linguaggio completamente client side, presente in quasi tutti i browser disponibili sul mercato e largamente supportato da Firefox.

Questo tipo di API ha qualche limitazione sia per quanto riguarda le funzioni disponibili sia per il numero di query che è possibile effettuare.

Per far funzionare l'applicazione si sono incontrate due tipologie di problemi.

La prima riguardava la problematica di far dialogare le tecnologie utilizzate, ovvero PHP e Javascript. Sostanzialmente il parser è effettuato lato client attraverso Javascript, l'applicazione richiede il nome della via ai Web Services di Via Michelin. Una volta ottenuta la risposta, deve passare queste informazioni all'applicazione in PHP che provvede a calcolare la velocità del tratto e memorizzare le informazioni nel database. Richiamare variabili PHP nel linguaggio Javascript non comporta particolari problemi, perché l'operazione viene effettuata lato server. L'operazione contraria, ovvero passare variabili Javascript generate lato client all'applicazione in PHP risulta un compito leggermente più complesso.

Per risolvere questo problema si è deciso di utilizzare SAJAX un toolkit open source basato sul framework AJAX, conosciuto anche come XMLHttpRequest. Attraverso SAJAX è possibile far dialogare i vari linguaggi, inoltre tramite le chiamate asincrone al web server non è necessario il refresh della pagina dell'applicazione [Mod].



Figura 4.9: SAJAX Toolkit

Il secondo problema era dettato dal fatto che le API gratuite permettono di effettuare un numero ridotto di query in un arco ristretto di tempo. La soluzione è stata quella di temporizzare le query, sostanzialmente il comando Javascript di alert, che visualizza una finestra lato browser, permette di avere un ritardo nella richiesta delle query. Inoltre è stato utilizzato il `SetTimeout` che permette di ritardare di x millisecondi l'esecuzione della funzione. In questo modo non si eccede il numero di query disponibili e l'applicazione funziona correttamente senza costi aggiunti per eventuali licenze.

4.4.3 DataBase

L'applicazione quindi riceve le informazioni sul nome della strada e le coordinate iniziali e finali del tratto percorso. Utilizza la stessa funzione, discussa nel paragrafo precedente, per calcolare la velocità ed effettua un inserimento nel database.

La query effettuata è la seguente :

```
"INSERT INTO strade VALUES('$ via ', $velocità ,NOW())"
```

La tabella del database ha una struttura molto semplice, essa è composta da tre campi :

- Nome della via di tipo TEXT
- Velocità di tipo DOUBLE
- Istante della rilevazione di tipo TIME

Tale tabella conterrà quindi tutte le rilevazioni inviate dai client. Anche se la struttura del DB è molto semplice e non sono effettuate operazioni dispendiose in fase di interrogazione, si potrebbero avere problemi legati alle dimensioni della tabella in particolari orari della giornata.

Le osservazioni presenti in database sono rimosse dopo un intervallo di ore specificato. La velocità media non sarà calcolata attraverso una media aritmetica. Tanto più una rilevazione è vicina nel tempo tanto più inciderà nel calcolo. Inoltre non dovrebbero concorrere al calcolo della velocità media tutti quei valori che si discostano molto dalla media osservata, in quanto potrebbero riportare un'osservazione non fedele. I problemi che potrebbero insorgere sono svariati:

- Il dispositivo client si potrebbe accidentalmente spegnere, come nell'esempio precedentemente discusso, riportando indicazione errate.

- Le vetture potrebbero viaggiare a velocità non consentite per legge, in questo modo altererebbero l'indicazione di velocità media.

In ogni caso l'utilizzo dell'applicazione nel tempo potrebbe aiutare a stimare la velocità media attesa su una determinata via. Così facendo si potrebbero subito individuare situazioni anomale. Inoltre attraverso queste informazioni si potrebbero suddividere le vie, per esempio una strada potrebbe essere catalogata come a scorrimento veloce, medio o lento.

Alcuni studi sulle reti VANET hanno dimostrato che basta equipaggiare anche un numero esiguo di autovetture (5-10% ma dipende dal contesto stradale) con il dispositivo utilizzato per avere una stima abbastanza precisa sullo stato di viabilità della strada.

4.4.4 I servizi offerti

Attualmente è possibile richiedere lo stato di viabilità di una particolare strada. Attraverso un form in HTML è possibile indicare il nome della via di interesse.



Figura 4.10: Richiesta info stato viabilità

Nel momento in cui viene sottomessa la richiesta (attraverso il bottone di SUBMIT del form) l'applicazione effettua una query nel database

individuando tutte le rilevazioni fornite dai client per quella determinata via.

In risposta il programma fornisce il numero di rilevazioni presenti in DB, la velocità media ed in funzione di quest'ultimo parametro colora il semaforo (giallo, rosso o verde) ereditando le stesse regole precedentemente illustrate per il componente CLIENT.

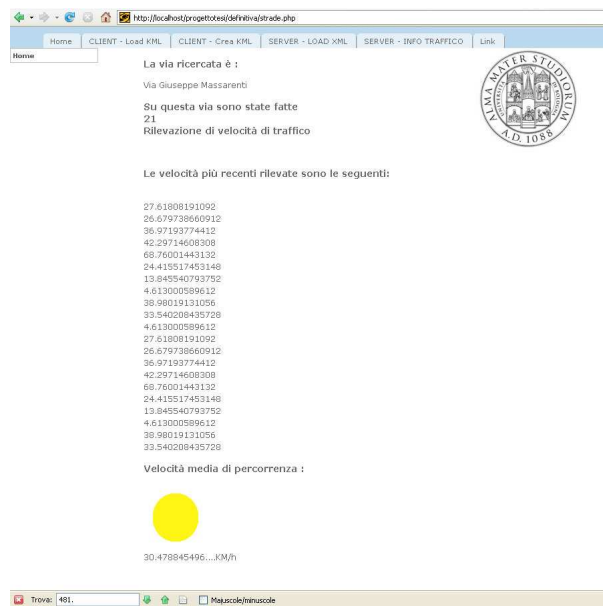


Figura 4.11: Risposta del server su stato viabilità

Un possibile scenario futuro del servizio è quello legare la velocità media delle strade ad un percorso richiesto da client. Nelle attuali applicazioni GPS quando un utente richiede un percorso da x a y, l'applicazione mostra sempre il percorso più breve in termini di tempo non tenendo conto dello stato di viabilità.

Attraverso le API gratuite è possibile visualizzare solo il percorso più breve, inoltre viene restituito un documento in HTML che è difficilmente manipolabile per gli scopi dell'applicazione discussa. Quindi è risultato dif-

ficile legare le informazioni di velocità delle strade ad un possibile cammino da x a y .

4.5 Sviluppi futuri

Lo sviluppo dell'applicazione è ancora in una fase embrionale quindi sono molti gli aspetti che si potranno migliorare in futuro.

Viene infatti tenuto traccia solo del nome della strada, quindi la stima della velocità è effettuata in base alle rilevazioni effettuate su tutta la via di interesse. Nel caso di vie molto estese, l'indicazione fornita risulta poco precisa e per niente indicativa dello stato attuale di viabilità.

Una soluzione potrebbe essere quella di segmentare le vie, anche in modo grossolano, attraverso il numero civico. Così facendo si riuscirebbero a fornire maggiori indicazioni specialmente per vie molto lunghe.

Le API che permettono di ottenere il dettaglio del numero civico sono a pagamento, questo farebbe aumentare i costi di realizzazione del progetto.

Attualmente non si riesce ad individuare la direzione di marcia del mezzo. Si potrebbe realizzare una funzione che riesca ad indicare

Questa caratteristica migliorerebbe di molto il servizio offerto, in quanto i veicoli hanno bisogno di conoscere la velocità di percorrenza ed eventuali informazioni addizionali attinenti la loro carreggiata.

In un futuro l'applicazione dovrebbe saper scegliere il percorso più conveniente in termini di tempo e risorse per muoversi da un punto x ad un punto y . L'algoritmo che determina questo percorso dovrebbe calcolare il percorso in funzione non solo della distanza minima tra i due punti ma tenendo anche in considerazione lo scenario complessivo della viabilità esistente nei tratti di strada presi in considerazione.

Capitolo 5

Conclusioni

Questo studio nasce dal ruolo sempre più importante che avranno le reti veicolari e le tecniche di comunicazione tra veicoli nei prossimi anni.

Tentando di prospettare i possibili scenari futuri, molteplici sono le funzionalità che le reti VANET potranno offrire:

- Informazioni per i viaggiatori. La diffusione di informazioni turistiche (ristoranti, musei, cinema...) o di servizio (ospedali, parcheggi...) nell'area attraversata dal veicolo. Pensiamo per esempio ad un servizio simile a quello fornito dalle pagine gialle/bianche, ovvero la ricerca di attività commerciali. L'utente che si troverà in una località a lui poco conosciuta potrebbe richiedere informazioni all'applicazione senza dover pagare costosissimi servizi telefonici.
- Monitoraggio ambientale. La rilevazione dei livelli di monossido di carbonio o di polveri sottili in una città è oggi demandato a centraline fisse dislocate in punti diversi della città. Se i rilevatori fossero installati su veicoli in servizio pubblico (autobus e taxi) e i risultati fossero

diffusi attraverso la Ivn in tempo reale, il monitoraggio si rivelerebbe più efficace.

- Integrazione di applicazioni legacy. Un unico box potrebbe includere funzioni svolte da dispositivi diversi che oggi popolano i parabrezza delle nostre auto, come il Telepass e i telecomandi di accesso a parcheggi aziendali o privati.
- Servizi meteo o di news. Solitamente queste informazioni vengono fornite in cambio della visione di messaggi pubblicitari. La posizione GPS potrebbe essere sfruttata per determinare il messaggio pubblicitario più idoneo. Tali attività promozionali stanno prendendo piede sul web negli ultimi mesi specialmente per filmati video.

Questa rassegna può fornire le basi per ulteriori studi sui protocolli utilizzati per la propagazione di messaggi di allerta che tengano conto anche dei differenti scenari in cui si applicano (es. ambienti urbani, extraurbani, autostradali ecc.).

Inoltre può fornire le basi per studi successivi sulla raccolta dati lato server. La gestione di dati potrebbe creare alcuni problemi a causa del numero crescente di dispositivi installati.

Molti sono gli aspetti di sicurezza ancora da definire data l'assenza di molte fasi di autenticazione non esistenti nell' 802.11p. Un ulteriore sviluppo del lavoro svolto può essere lo studio sistematico e di conseguenza l'individuazione di possibili soluzioni per arginare o neutralizzare attacchi al web server.

Appendice A

Proto-Codice Applicazione

A.1 Client

A.1.1 Load KML

```
1<?php
2// Questa funzione permette di caricare un file xml/kml.
3// Successivamente viene analizzato attraverso DOM
4// e con l'ausilio di altre funzioni
5// viene effettuato un inserimento in DB
6function distaste($pathkml){
7
8    $doc = new DOMDocument();
9    $doc->load($pathkml);
10
11    $books = $doc->getElementsByTagName( "coordinates" );
12    $fine = $books->item(1)->nodeValue;
13
14    $pieces = explode(",",$fine);
15    $conteggio = count($pieces)/3;
16
```

```
17 $primo =0;
18     for ($pluto=0;$pluto<$conteggio;$pluto++){
19
20     $a=0;
21     if($primo == 0){
22         $latitu1 = current($pieces);}
23     else{
24         $latitu1 = next($pieces);
25     }
26     if($latitu1 != null or $latitu1 != 0){
27         $a++;
28     };
29     $latitu2 = next($pieces);
30     $latitu0 = next($pieces);
31     if($latitu2 != null or $latitu2 != 0){
32         $a++;
33     };
34     $latitu3 = next($pieces);
35     if($latitu3 != null or $latitu3 != 0){
36         $a++;
37     };
38     $latitu4 = next($pieces);
39     if($latitu4 != null or $latitu4 != 0){
40         $a++;
41     };
42     $latitu00 = next($pieces);
43     echo($latitu1);
44     echo("<br>");
45     echo($latitu2);
46     echo("<br>");
47     echo($latitu0);
48     echo("<br>");
49     echo($latitu3);
```

```

50     echo("<br>");
51     echo($latitu4);
52     echo("<br>");
53     echo($latitu00);
54     echo("<br>");
55     echo($a);
56     echo("<br>");
57
58     if($a==4){
59
60         // Assumo che la frequenza delle rilevazioni sia di 10 secondi
61         // quindi devo moltiplicare il tutto per 360
62         $velocita= (distance($latitu4,$latitu3,$latitu2,$latitu1,"K") * 360 );
63
64         addsql($latitu4,$latitu3,$latitu2,$latitu1,$velocita);
65
66         prev($pieces);
67         prev($pieces);
68         prev($pieces);
69
70     }
71 $primo++;
72 }
73
74 }
75
76 // Questa funzione permette di effettuare l'inserimento in Database .
77 function addsql($l,$lo,$lf,$lof,$velo){
78
79     $miconnetto = mysql_connect('localhost','root','passw');
80     mysql_select_db('wireless');
81
82     $query="INSERT INTO client VALUES($l, $lo,$lf,$lof,$velo,NOW())";

```

```
83
84     mysql_query($query) or die(mysql_error());
85     mysql_close($miconnestto);
86 }
87 // Questa funzione permette di calcolare la distanza esistente
88 // tra 2 punti . Vengono immesse le coordinate GPS
89 // sia del punto iniziale che di quello finale.
90 // È possibile specificare la l'unità di misura
91 // della distanza K per i chilometri N per le miglia nautiche
92 // altrimenti viene restituito il valore in miglia.
93 function distance($lat1 , $lon1 , $lat2 , $lon2 , $unit) {
94
95     $theta = $lon1 - $lon2;
96     $dist = sin(deg2rad($lat1)) * sin(deg2rad($lat2)) +
97     cos(deg2rad($lat1)) * cos(deg2rad($lat2)) * cos(deg2rad($theta));
98     $dist = acos($dist);
99     $dist = rad2deg($dist);
100    $miles = $dist * 60 * 1.1515;
101    $unit = strtoupper($unit);
102
103    if ($unit == "K") {
104        return ($miles * 1.609344);
105    } else if ($unit == "N") {
106        return ($miles * 0.8684);
107    } else {
108        return $miles;
109    }
110 }
111 ?>
```

A.1.2 Crea KML

```
1 <?php
2
3 function creakml($latini , $longini , $latfin , $longfin , $nomefile){
4
5     mysql_connect( 'localhost' , 'root' , 'passw' );
6     mysql_select_db( 'wireless' );
7
8     $doc = new DOMDocument( '1.0' , "ISO-8859-1" );
9     $doc->formatOutput = true;
10
11     $root = $doc->createElement( 'kml' );
12     $root = $doc->appendChild( $root );
13
14     $root->setAttribute( "xmlns" , "http://www.google.com/earth/kml/2" );
15
16     $title = $doc->createElement( 'Document' );
17     $title = $root->appendChild( $title );
18
19
20     $name = $doc->createElement( 'name' );
21     $name = $title->appendChild( $name );
22
23     $testoname = $doc->createTextNode( "Tesi reti Wireless" );
24     $name->appendChild( $testoname );
25
26     $stile = $doc->createElement( 'Style' );
27     $stile = $title->appendChild( $stile );
28
29     $stile->setAttribute( "id" , "rosso" );
30
31     $icos = $doc->createElement( 'LineStyle' );
```

```
32     $icos = $stile->appendChild($icos);
33
34
35     $ico = $doc->createElement('color');
36     $ico = $icos->appendChild($ico);
37
38     $ctext = $doc->createTextNode('7f000ff');
39     $ctext = $ico->appendChild($ctext);
40
41     $hfer = $doc->createElement('width');
42     $hfer = $icos->appendChild($hfer);
43
44     $text = $doc->createTextNode('6');
45     $text = $hfer->appendChild($text);
46
47     $stile1 = $doc->createElement('Style');
48     $stile1 = $title->appendChild($stile1);
49
50     $stile1->setAttribute("id", "giallo");
51
52     $icosgiallo = $doc->createElement('LineStyle');
53     $icosgiallo = $stile1->appendChild($icosgiallo);
54
55
56     $icoloreg = $doc->createElement('color');
57     $icoloreg = $icosgiallo->appendChild($icoloreg);
58
59     $ctextgial = $doc->createTextNode('ff00ffff');
60     $ctextgial = $icoloreg->appendChild($ctextgial);
61
62     $hfergia = $doc->createElement('width');
63     $hfergia = $icosgiallo->appendChild($hfergia);
64
```

```

65     $textlargial = $doc->createTextNode('6');
66     $textlargafile=$hfergia->appendChild($textlargial);
67
68     $stile2 = $doc->createElement('Style');
69     $stile2 = $title->appendChild($stile2);
70
71     $stile2->setAttribute("id", "verde");
72
73     $icosverde = $doc->createElement('LineStyle');
74     $icosverde = $stile2->appendChild($icosverde);
75
76
77     $icolorev = $doc->createElement('color');
78     $icolorev = $icosverde->appendChild($icolorev);
79
80     $ctextverd = $doc->createTextNode('7f10ff3d');
81     $ctextverd = $icolorev->appendChild($ctextverd);
82
83     $hferverd = $doc->createElement('width');
84     $hferverd = $icosverde->appendChild($hferverd);
85
86     $textlarverd = $doc->createTextNode('6');
87     $textlargafile=$hferverd->appendChild($textlarverd);
88
89     //Elimino tutte le osservazioni relative a tutti i giorni precedenti
90     $anno = date("Y");
91     $mese = date("m");
92     $gg = date("d")-1;
93
94
95     $query2="DELETE FROM 'client' WHERE 'time' <= '$anno-$mese-$gg'";
96     mysql_query($query2)or die(mysql_error());
97

```

```
98     $query="SELECT * FROM client WHERE
99     (lat BETWEEN $latini AND $latfin) AND
100    (lon BETWEEN $longini AND $longfin)";
101    $result2= mysql_query($query)or die(mysql_error());
102
103    echo(mysql_num_rows($result2));
104    print("<br>");
105    for ($nres=mysql_num_rows($result2); $nres > 0; $nres --){
106
107        print("<br>");
108        $namearr2 = mysql_fetch_object($result2);
109
110        $gulp = $namearr2->vel;
111
112        $colore = "";
113        if ($gulp < 20){
114            $colore = "rosso";
115        }elseif($gulp < 40){
116            $colore = "giallo";
117        }else
118        {$colore = "verde";}
119
120        $place = $doc->createElement('Placemark');
121        $place = $title->appendChild($place);
122
123
124        $surl = $doc->createElement('styleUrl');
125        $surl = $place->appendChild($surl);
126
127        $colore = $doc->createTextNode("#$colore");
128        $surl->appendChild($colore);
129
130        $sline = $doc->createElement('LineString');
```

```
131     $sline = $place->appendChild($sline);
132
133     $coo = $doc->createElement('coordinates');
134     $coo = $sline->appendChild($coo);
135
136     $testo = $doc->createTextNode("$namearr2->lon,$namearr2->lat");
137     $coo->appendChild($testo);
138
139     $testo1 = $doc->createTextNode(",0,");
140     $coo->appendChild($testo1);
141
142     $testo2 = $doc->createTextNode("$namearr2->longf,$namearr2->latf");
143     $coo->appendChild($testo2);
144
145     $testo3 = $doc->createTextNode(",0");
146     $coo->appendChild($testo3);
147
148     }
149     $doc->save($nomefile);
150 }
151 ?>
```

A.2 Server

A.2.1 Reverse Geocoding

```
1 <?
2 // Richiede il file Sajax per gestire le richieste asincrone
3 // in Ajax tra html javascript php
4 require("Sajax.php");
5
6 // Inserisce nel Database il nome della via del punto finale
7 // e la velocità di percorrenza
8 function inseriscivia($x,$velo) {
9
10     $miconnestto = mysql_connect('localhost','root','passw');
11     mysql_select_db('wireless');
12
13     $query="INSERT INTO strade VALUES('$x',$velo,NOW())";
14
15     mysql_query($query)or die(mysql_error());
16     mysql_close($miconnestto);
17
18 }
19
20 sajax_init();
21 // Lista delle funzioni da esportare
22 sajax_export("inseriscivia","distance");
23 // Istanza del client
24 sajax_handle_client_request();
25
26 ?>
27 <html>
28     <script >
29     <?>
```

```
30     sajax_show_javascript ();
31     ?>
32     function set_math_result(result) {
33
34     }
35     function do_the_math(a,b) {
36         x_inseriscivia(a,b,set_math_result);
37     }
38     </script>
39
40
41 <!-- API VIA MICHELIN Per effettuare il REVERSE GEOCODING-->
42 <script src="http://api.viamichelin.com/apijs/js/api.1.0.js"></script>
43 <script>VMAPI.registerKey("JSBS20070201173627379646361061");</script>
44 <script>
45 // Variabile globale utilizzata per memorizzare la velocità di percorrenza
46 var velo= 0;
47
48 function pluto(z){
49     velo = 0;
50     velo = z*360;
51 }
52
53 // La funziona prende in input latitudine e longitudine dei due punti
54 // calcola la velocità di percorrenza , ricerca la via del punto finale
55 // Attraverso le funzioni chiamate fa un inserimento in DB
56 function functiongeocodesearch(lai ,loi ,laf ,lof) {
57
58     lat = parseFloat(lai);
59     long = parseFloat(loi);
60
61     latf = parseFloat(laf);
62     longf = parseFloat(lof);
```

```
63
64     alert ("inserisce strada in DB");
65
66     x_distance(lai ,loi ,laf ,lof ,"K" ,pluto);
67
68     reversegeo = new VMReverseGeo();
69     mylonlat = new VMLonLat(latf ,longf);
70
71     reversegeo.addEventHandler("onCallBack" ,functiondisplayresult);
72     reversegeo.search(mylonlat);
73 }
74
75 function functiondisplayresult() {
76
77     myaddress = reversegeo.result;
78
79     var para = document.getElementById("to");
80     para.lastChild.nodeValue =
81     para.lastChild.nodeValue+myaddress.address;
82
83     var ciccio = myaddress.address;
84     do_the_math(ciccio ,velo);
85
86 }
87 var xmlDoc;
88
89 // Questa funzione gestisce correttamente il caricamento
90 // dei file XML in Javascript
91 function loadXML()
92 {
93 //Carica il file xml
94 // codice per IE
95 if (window.ActiveXObject)
```

```
96 {
97     xmlDoc=new ActiveXObject("Microsoft.XMLDOM");
98     xmlDoc.async=false;
99     xmlDoc.load("<? print $_FILES['percorso']['name']?>");
100     getmessage();
101 }
102 // codice per Mozilla, Firefox, Opera, etc.
103 else if (document.implementation && document.implementation.createDocument)
104 {
105
106     xmlDoc=document.implementation.createDocument("", "", null);
107     xmlDoc.load("<? print $_FILES['percorso']['name']?>");
108     xmlDoc.onload=getmessage;
109 }
110 else{
111     alert('Your browser cannot handle this script');
112 }
113 }
114
115 // Estrae informazioni dal file XML caricato
116 function getmessage(){
117
118     risultato = xmlDoc.getElementsByTagName
119     ("coordinates")[1].childNodes[0].nodeValue;
120     elabora(risultato);
121
122 }
123
124 function elabora(pluto) {
125
126     var s=pluto;
127     var out = new String;
128     var nomi = new Array();
```

```
129
130     nomi = s.split(",0,");
131     out = "Trovati " + nomi.length + "Lunghezze percose .\n";
132     out += "Eccoli:\n";
133     a = ')';
134     for (i=0;i<nomi.length;i++) {
135         var plutino = 'functiongeocodesearch('+nomi[i]+' ,'+nomi[i+1]+a ;
136             setTimeout(plutino,2000);
137             out += i + ") " + nomi[i] + "<br \>";
138             i++;
139     }
140
141     document.getElementById("pu").innerHTML = out;
142
143 };
144
145 function creamappa(){
146
147     map = new VMMap(document.getElementById("yourmapdiv"));
148     map.drawMap(mylonlat,11);
149 }
150
151 </script>
152 </html>
```

A.2.2 Info Viabilità

```

1 <?
2 function infostrada($via){
3
4     $miconnetto = mysql_connect('localhost','root','passw');
5     mysql_select_db('wireless');
6     $query = "SELECT * FROM strade WHERE via = '$via' " ;
7
8     $result2= mysql_query($query)or die(mysql_error());
9     print("<h3> Su questa via sono state fatte <br />");
10    echo(mysql_num_rows($result2));
11    print("<br />Rilevazione di velocità di traffico </h3>");
12    print("<br>");
13
14
15    print("<h3>Le velocità più recenti rilevate sono le seguenti:</h3>");
16    $velm = 0;
17    for($nres=mysql_num_rows($result2);$nres>0;$nres--){
18
19        print("<br>");
20        $namearr2 = mysql_fetch_object($result2);
21        $gulp = $namearr2->velocita;
22        $velm += $gulp;
23
24        print($gulp);
25    }
26
27    if(mysql_num_rows($result2) !== 0){
28
29        $velm = $velm / mysql_num_rows($result2);
30
31        if ($velm <20){

```

```
32             $colore = "img/rosso.png";
33     }elseif($velm<40){
34             $colore = "img/giallo.png";
35     }else
36     {$colore = "img/verde.png";}
37
38     echo("<h3> Velocità media di percorrenza :</h3>
39     <img src=$colore /><br />");
40     print("$velm."...KM/h");
41
42 }
43
44
45 }
46 ?>
```

Bibliografia

- [AE03] Matthias Lott Rüdiger Halfmann André Ebner, Hermann Rohling. Decentralized slot synchronization in highly dynamic ad hoc networks. Technical report, Technical University of HamburgHarburg, 2003.
- [Ben04] A. Benslimane. *Optimized Dissemination of Alarm Messages in Vehicular Ad-Hoc Networks (VANET)*, volume 3079. Springer Berlin, Heidelberg, Laboratoire d Informatique d Avignon LIA France, 2004.
- [Cen] Hexa Software Development Center. Geographical distance calculations. <https://www.zipcodeworld.com/docs/distance.pdf>.
- [CM99] S. Corson and J. Macker. Mobile Ad hoc Networking (MANET): Routing Protocol Performance Issues and Evaluation Considerations. RFC 2501 (Informational), January 1999.
- [EF05] Andrea Zanella Elena Fasolo, Roberto Furiato. Smart broadcast algorithm for inter vehicular communications. Technical report, Università degli studi di Padova, 2005.
- [Gar] Garmin. Introduzione gps. <http://www.garmin.com/about-GPS/manual.html>.

- [Goo] Google. What is kml? <http://code.google.com/apis/kml/>.
- [gps] Gps history, chronology, and budgets. <http://www.rand.org/pubs/monograph-reports/MR614/MR614.appb.pdf>.
- [GS] Mesut Günes and Otto Spaniol. Routing algorithms for mobile multi-hop ad-hoc. <http://citeseer.ist.psu.edu/566709.html>.
- [HK05] Ali Hamidian and Ulf Korner. Enhancing the ieee 802.11e edca to provide qos guarantees. Technical report, Department of Communication Systems, Lund University, 2005.
- [IBM07] IBM. Tuning lamp systems, part 1-2-3. <http://www.ibm.com/developerworks/linux/library/l-tune-lamp-1/index.html>, 2007. [Online; accessed 10-novembre-2007].
- [Ipe] Ipersec.com. Benchmarking php accelerators. <http://www.ipersec.com/index.php/2006/05/30/benchmarking-php-accelerators/>.
- [KV00] Young-Bae Ko and Nitin H. Vaidya. Location-aided routing (lar) in mobile ad hoc networks. Technical report, Department of Computer Science, Texas AM University, 2000.
- [LS06] Yunpeng Zang Lothar Stibor. Neighborhood evaluation of vehicular ad-hoc network using ieee 802.11p. Technical report, ComNets RWTH-Aachen University, 2006.
- [MEZ03] Gene Tsudik Nalini Venkatasubramanian Magda El Zarki, Sharad Mehrotra. Security issues in a future vehicular network.

Technical report, Department of Information and Computer Science, University of California, Irvine, 2003.

- [Mod] ModernMethod. Ajax toolkit for php - sajax - simple ajax toolkit.
- [Muc06] Alberto Muccil. Il progetto galileo sta diventando realtà, 2006.
- [Mum03] P.J. Mumfordl. Relative timing characteristics of the one pulse per second (1pps) output pulse of three gps receivers, 2003.
- [OMWE07] G. Orfanos, J. Mirkovic, B. Walke, and S. Emmadi. A centralized mac protocol with qos suport for wireless lans. In *Proceedings of International Symposium on Personal, Indoor and Mobile Radio Communications (PIRMC) 2007*, page 5, Athens, Greece, Sep 2007. IEEE Communication Society.
- [Pig04] Riccardo Pighil. La tecnica di trasmissione ofdm, 2004.
- [Pol06] Daniele Polini. *Protocolli di assegnazione degli indirizzi IP in reti Manet: progettazione e valutazione*. PhD thesis, Università degli studi di Pisa, 2006.
- [SPKW07] M. Schinnenburg, R. Pabst, K. Klagges, and B. Walke. A software architecture for modular implementation of adaptive protocol stacks. In *MMBnet Workshop 2007*, pages 94–103, Hamburg, Germany, Sep 2007. University of Hamburg, Computer Science Department, TKRN - Telecommunication and Computer Networks Group, Wolfinger, Bernd and Heidtmann, Klaus.

- [Via07] ViaMichelin. Maps drive api - viamichelin btob, 2007. [Online; accessed 10-novembre-2007].
- [Wik07a] Wikipedia. Lamp software bundle— wikipedia, l'enciclopedia libera, 2007. [Online; accessed 10-novembre-2007].
- [Wik07b] Wikipedia. Sistema di posizionamento galileo— wikipedia, l'enciclopedia libera, 2007. [Online; accessed 10-novembre-2007].
- [WR06] Lars Wischhof and Hermann Rohling. Congestion control in vehicular ad hoc networks. Technical report, Hamburg University of Technology, 2006.
- [WXG06] Paul Richardson Weidong Xiang and Jinhua Guo. Introduction and preliminary experimental results of wireless access for vehicular environments (wave) systems. Technical report, University of Michigan, 2006.
- [YZ07] Hari Balakrishnan Samuel Madden Yang Zhang, Bret Hull. Icedb: Intermittently-connected continuous query processing. Technical report, MIT Computer Science and Artificial Intelligence Laboratory, 2007.