POLITECNICO DI TORINO

Corso di Laurea in Ingegneria Informatica

Tesi di Laurea Magistrale

Analisi di sicurezza di servizi sviluppati su reti Vanet

Relatori
prof. Antonio Lioy
prof. Cataldo Basile

Candidato
Angelo Bruccoleri

Anno accademico 2016-2017

Sommario

La *Vehicular Ad Hoc Network* (VANET) è stata oggetto nel corso dell'ultimo decennio di una crescente attenzione sia da parte dell'ambito accademico che dell'industria automotive grazie all'importante e ormai noto obiettivo prefissato ovvero quello di rendere più sicuri gli spostamenti veicolari sulle strade di tutto il mondo. VANET è una tecnologia emergente ed è una speciale classe di rete MANET (*Mobility Ad Hoc Network*) in cui i nodi che si scambiano le informazioni sono rappresentati da veicoli in movimento su dei "percorsi" obbligati definiti dalla conformazione dell'infrastruttura stradale sulla quale si trovano. Affinché questa interessante tecnologia possa diventare una realtà tangibile, una serie di ostacoli devono essere affrontati e discussi, a partire dall'organizzazione dell'infrastruttura di rete e dagli standard che definiscono i canali di comunicazione da utilizzare per lo scambio dati. Negli ultimi anni il campo di maggiore interesse sul quale si sono concentrati gli sforzi maggiori è stato senza dubbio quello della *sicurezza* e in tal senso un fattore da non sottovalutare è la *privacy dei nodi*. Ogni utente che vorrà fare affidamento alla rete VANET e ai suoi servizi dovrà vedersi garantita la propria privacy onde evitare situazioni spiacevoli che possano ad esempio condurre a dei furti di informazioni sensibili o a dei tracciamenti degli spostamenti. La sensibilità di questa problematica ha permesso di definire diversi schemi di protezione che potessero offrire agli utilizzatori finali la giusta arma di difesa verso i cosiddetti attaccanti e una soluzione valida, come poi verrà spiegato nel corso della trattazione, si basa sull'utilizzo dei cosiddetti *pseudonimi*.

Securizzare la rete VANET è un obiettivo comune a tutte le parti interessate che hanno speso tempo e denaro nella speranza di poter sviluppare delle soluzioni che potessero consentire all'utilizzatore finale di poter fruire e sfruttare a pieno le potenzialità e le funzionalità offerte da questa tipologia di rete. Le applicazioni di sicurezza richiedono che le informazioni sensibili e real-time vengano scambiate periodicamente tra tutti i nodi della rete attraverso dei messaggi avente una struttura interna ben definita: i *beacon*. Oltre alle applicazioni di sicurezza diversi altri servizi connessi al comfort degli utenti possono vedere il loro impiego in queste reti. Basti ad esempio pensare ai servizi che distribuiscono le informazioni relative al meteo o alle condizioni della viabilità stradale in tempo reale o ancora allo streaming di dati

audio/video. Un ulteriore concetto di assoluto rilievo da tenere in seria considerazione è la *disponibilità* della rete. Poiché le informazioni "critiche" devono fluire attraverso essa, è certamente importante evitare che utenti non autorizzati possano, mediante dei comportamenti inappropriati, causare dei malfunzionamenti e mettere in grave pericolo gli stessi utenti che affidano a questa stessa rete la propria "vita". Infatti limitando la disponibilità della rete quelle informazioni spesso indicate come "*time sensitive information*" potrebbero non giungere in tempo alla destinazione e causare dei gravi disagi o addirittura incidenti che metterebbero in forte dubbio la vera utilità di questa rete. Purtroppo la VANET può avere molti punti deboli sfruttabili dai cosiddetti attaccanti per limitarne le funzionalità e questo è il motivo principale per cui ancora oggi non si è giunti ad una vera e propria applicazione reale di questa tecnologia di assoluta importanza. A tal proposito nel corso della trattazione verrà effettuata una classificazione dettagliata delle varie classi di attaccanti, dei requisiti di sicurezza richiesti da una Vehicular Ad Hoc Network e dei possibili attacchi e relative contromisure (se presenti).

L'obiettivo della presente tesi è quello di analizzare e valutare la sicurezza di questa tipologia di rete e dei servizi attivabili partendo dalla pianificazione di una simulazione su uno scenario di traffico reale nella Città di Bologna mediante l'utilizzo di un noto simulatore di traffico chiamato SUMO e continuando con lo sviluppo di un'applicazione Java, chiamata *SimulBeaconXchange*, che simula lo scambio di beacon tra tutti i nodi della rete tra i quali sono stati appositamente inseriti degli attaccanti. Essi, attraverso i beacon generati ed immessi sulla rete, consentiranno nella fase finale di impostare delle simulazioni ad hoc per valutare l'impatto sulla rete delle diverse tipologie di attacco e di delineare le possibili strategie per rilevarli, mitigarli e possibilmente combatterli.

Ringraziamenti

Desidero spendere qualche parola in favore di tutte quelle persone che mi hanno consentito di portare a compimento questo percorso di studio e che mi hanno aiutato, ognuno in modo diverso, con le loro critiche, con il loro supporto o semplicemente con qualche parola di incoraggiamento.

Un sentito ringraziamento va a chi mi ha dato l'opportunità di trattare questa importantissima tematica ovvero il Professor Antonio Lioy ed il Professor Cataldo Basile, rispettivamente relatore e co-relatore della presente tesi, perché grazie alla loro professionalità, alle loro conoscenze ed al loro aiuto ho potuto condurre questo studio nel modo più lineare possibile.

Un Grazie immenso va ai miei genitori che mi hanno sostenuto in ogni senso in questa "avventura" permettendomi di raggiungere questo traguardo tanto desiderato, alla mia famiglia che mi ha sempre supportato anche nei momenti più difficili, agli amici che in questi anni hanno dovuto sopportare me e le mie incertezze e per ultimo, ma non per ultima, alla mia fidanzata che nonostante il peso della lontananza e nonostante abbia dovuto subire probabilmente più di chiunque altro ogni mia scelta ed ogni mio momento di difficoltà ha sempre trovato la forza per incoraggiarmi e sostenermi.

Grazie di cuore a tutti quanti.

Indice

Capitolo 1

Introduzione

Al giorno d'oggi il mezzo di trasporto preferito dalle persone di tutto il mondo per gli spostamenti terreni è l'autoveicolo. Purtroppo risulta ancora molto elevato il numero di incidenti che ogni giorno causano morti sulle strade e, al fine di contrastare questo fenomeno, le case automobilistiche hanno sviluppato negli ultimi anni diversi sistemi di sicurezza attiva a bordo dei veicoli ormai di serie che possono sicuramente aiutare ad evitare incidenti talvolta fatali.

Unitamente a questi sistemi di sicurezza, stanno prendendo sempre più piede e stanno sempre più sviluppandosi i cosiddetti sistemi di infotainment che mirano a far diventare il veicolo stesso un oggetto "smart" capace di offrire servizi di connettività ed applicazioni ad hoc fino a qualche anno fa disponibili solo per i dispositivi mobili come smartphone o tablet. Questi sistemi possono indubbiamente essere sfruttati per l'implementazione di un futuro standard di comunicazione inter-veicolare che consenta la trasmissione e la condivisione di informazioni che permettano, ad esempio, di segnalare una situazione di pericolo presentatasi in un certo momento sull'asse stradale. Si potrebbe pensare di estendere questa idea e consentire la trasmissione di semplici indicazioni utili per il comfort dei viaggiatori come informazioni sul meteo, sulla viabilità, sulla presenza di stazioni di rifornimento o ristoranti nelle immediate vicinanze del veicolo stesso.

Oggigiorno, il bisogno degli utenti di avere un accesso ad Internet in qualunque posto si trovino ed in qualsiasi momento della giornata sta diventando una necessità. A partire da questa considerazione si potrebbe quindi pensare di offrire servizi di connettività a bordo dei veicoli mediante applicazioni specifiche che permettano, ad esempio, lo scambio di dati tra nodi limitrofi o il download di file dalla rete o ancora lo streaming audio/video. L'idea appena citata oggi ha un nome: VANET (Vehicular Ad-hoc NETwork). La maggiore espressione e applicazione delle reti VANET è l'ITS (Intelligent Transportation System).

Si prevede che nel prossimo futuro le automobili diventeranno parte integrante della rete internet sia come end-point mobili che come sensori mobili. A tal proposito anche le principali case automobilistiche come Audi, FCA, Renault hanno investito molto in questo campo negli ultimi anni dando, tra le altre cose, vita all'organizzazione C2CCC (Car-2-Car Communication Consortium).

L'obiettivo principale delle VANET è quello di creare un'infrastruttura di rete costituita solamente da nodi mobili (veicoli) senza l'aiuto di una stazione centrale o

controllore esterno che ne coordini il funzionamento. La topologia della rete non è statica ma è in continua evoluzione a causa della mobilità dei suoi nodi.

Nell'ultimo decennio è comunque cresciuto molto l'interesse da parte della comunità scientifica verso queste reti e molti sono stati gli esperimenti e le pubblicazioni fatte da diversi gruppi di ricerca in ambito accademico e non, in diversi parti del mondo, al fine di investigare su ogni singolo aspetto di una rete di questo tipo: dalle possibili applicazioni all'architettura di rete fino alla privacy nelle comunicazioni. Con il termine privacy si suole indicare in genere il concetto di "comunicazione sicura tra due o più entità". Nel contempo, con il termine sicurezza si fa riferimento al concetto di accuratezza delle informazioni scambiate tra le diverse entità e di affidabilità verso coloro i quali hanno generato queste stesse informazioni. Queste sono solo alcune delle problematiche che la presente tesi si propone di affrontare e discutere per la definizione di una soluzione che possa offrire una maggiore protezione agli utenti di queste reti.

La sicurezza delle comunicazioni viene ancor prima dello studio e dello sviluppo delle applicazioni che possono essere attivate su questo tipo di reti in quanto è un prerequisito fondamentale per un corretto funzionamento delle VANET. Un attaccante, ad esempio, potrebbe introdurre informazioni false sulla rete, conoscere la posizione di un certo nodo all'interno della rete, tracciarne il percorso o prelevare dati sensibili scambiati attraverso i beacon (messaggi) dai nodi mobili.

Nelle pagine a seguire è possibile trovare una breve descrizione delle principali proprietà delle VANET fino a raggiungere più avanti il vero punto di interesse di questa trattazione: lo studio e l'analisi della sicurezza delle applicazioni e dei servizi sviluppati sulle Vehicular Ad Hoc Network.

1.1 Struttura della tesi

Il lavoro di tesi è strutturato in 7 Capitoli e 2 Appendici (A e B). Questo primo capitolo comprende una breve descrizione dell'ambito di studio nel quale si colloca la presente tesi utile per introdurre il lettore nel mondo delle VANET.

Il capitolo 2 affronta diverse tematiche connesse alle Vehicular Ad Hoc Network considerando caratteristiche, proprietà ed algoritmi sviluppati per la loro implementazione. Viene inoltre fatto un accenno alle componenti costruttive, ai protocolli di routing e agli standard fino ad oggi definiti.

Il capitolo 3 si concentra su un concetto cardine delle VANET: il beaconing, ovvero il processo di scambio di messaggi tra i veicoli. Vengono quindi fatte delle considerazioni e delle osservazioni sull'importanza del beaconing e sulle problematiche relative alla sicurezza di questo processo, con riferimento ai lavori e ai risultati ottenuti dalle ricerche condotte nel corso degli ultimi anni.

Il capitolo 4 introduce e quindi descrive minuziosamente il problema della sicurezza nelle reti VANET a partire dai requisiti e dalle proprietà di sicurezza definiti dallo standard X.800. A seguire viene proposta una classificazione delle diverse tipologie di attaccanti, delle varie forme di attacco che possono essere condotte su questo tipo di reti e delle conseguenze di questi attacchi con riferimenti ai vari scenari possibili

e alle soluzioni proposte per la loro mitigazione. Infine viene posta l'attenzione sul problema relativo alla privacy delle varie entità che fanno parte dell'architettura di rete considerando il concetto di "pseudonimo".

Nel capitolo 5 viene mostrata la prima parte dello studio condotto a partire dalla simulazione di uno scenario di traffico reale con SUMO e dalla "progettazione" delle tabelle sulla base di dati che dovranno contenere le informazioni, anch'esse minuziosamente descritte, elaborate ai vari step del processo. Inoltre verrà fatta una breve descrizione dell'applicazione *SimulBeaconXchange* appositamente sviluppata per simulare lo scambio di messaggi tra i veicoli e creare l'ambiente virtuale sul quale poter sperimentare le soluzioni sviluppate e, per concludere, sarà definito l'attack model del sistema trattato ed affrontato il problema del logging dei messaggi sulle memorie delle OBU installate a bordo dei veicoli.

Il capitolo 6 continua il focus sull'applicazione SBX per metterne in evidenza le componenti di basso livello più interessanti e riprende, approfondendolo, il problema della sicurezza del processo di beaconing mettendo in evidenza gli aggiornamenti apportati recentemente allo standard IEEE 1609. Successivamente vengono "rivalutate" le varie tipologie di attacco al fine di trarne spunto per la definizione e la simulazione di possibili scenari utili per lo studio e la ricerca di eventuali soluzioni che possano proteggere la rete da potenziali malintenzionati.

Infine, il capitolo 7 conclude l'intera trattazione con una sintesi del lavoro svolto e dei risultati ottenuti oltre che con una breve discussione sui possibili lavori futuri connessi alla tematica studiata nella presente tesi e all'applicazione sviluppata. Successivamente, a concludere il tutto, sarà presente una breve parentesi che raccoglie le considerazioni e le valutazioni personali relative a questa importantissima tecnologia.

Gli appendici A e B contengono rispettivamente il Manuale del Programmatore ed il Manuale Utente con le descrizioni degli script utilizzati e dei passaggi da effettuare per un uso corretto del progetto presentato.

Capitolo 2

VANET: proprietà e caratteristiche

Agli inizi degli anni 2000 le reti VANET erano viste come una semplice applicazione dei principi già noti delle reti MANET (*Mobility Ad-hoc NETwork*) dalle quali però si differenziano per architettura costruttiva, caratteristiche ed applicazioni. Solo successivamente il termine VANET è saltato fuori con prepotenza e viene oggi usato come sinonimo del più generico concetto di IVC (*Inter-Vehicle Communication*).

Nel corso degli ultimi anni sono stati proposti diversi standard in diverse parti del globo terrestre a testimonianza dell'interesse posto su questa nuova applicazione. In particolare si tratta di Giappone (*ARIB STD-T109*), USA (*IEEE 1609*) ed Europa (*ETSI ITS G5*).

Volendo fornire una definizione di rete VANET possiamo dire che *"si tratta di un tipo particolare di rete MANET nella quale i nodi mobili sono rappresentati dai veicoli in movimento sull'asse stradale"*. Uno dei principali obiettivi di questa tipologia di rete è quello di aumentare il livello di sicurezza stradale e migliorare il comfort di viaggio degli utenti sulle strade di tutto il mondo. Una VANET, come è possibile vedere dalla Figura 2.1, è una forma di wireless ad-hoc network costituita da un insieme di veicoli che fungono da nodi aventi capacità di comunicazione sia con i veicoli vicini (comunicazione Vehicle-to-Vehicle o V2V) e sia con le cosiddette RSU (comunicazione Vehicle-to-Infrastructure o V2I). Ogni nodo si comporta sia da router che da host. Da ciò si evince che non esiste una infrastruttura di rete vera e propria ma la comunicazione avviene grazie alla capacità di ogni veicolo di ricevere, elaborare ed inoltrare i messaggi grazie alle cosiddette OBU installate a bordo del veicolo stesso. Questo è il motivo per cui una VANET è definita rete "infrastructure-less" (letteralmente senza infrastruttura) o multi-hop. Ciò giustifica il comportamento dinamico di queste reti: la topologia e il quantitativo di dati immessi su una parte della rete dipende dalla densità dei veicoli presenti in un certo istante di tempo in quello specifico punto della rete.

Oltre alla topologia, anche la connettività nelle VANET potrebbe cambiare spesso e specialmente quando la densità dei veicoli è bassa, si ha una maggiore probabilità di disconnessione dalla rete. In alcune applicazioni, come l'accesso a Internet, il problema deve essere considerato e risolto. Una possibile soluzione è quella di installare diversi ripetitori o Access Point lungo la strada per mantenere la connettività.

Uno dei possibili ostacoli da considerare per una corretta comunicazione è la "gestione" dei messaggi scambiati. Ci si riferisce a questo processo con i concetti di *Information Dissemination* ed *Information Aggregation*. Il primo ha l'obiettivo di mantenere in vita una certa informazione sulla rete per un certo periodo di tempo e renderla quindi disponibile anche ai veicoli che arriveranno successivamente mentre il secondo fa riferimento al processo di elaborazione e memorizzazione dei messaggi da parte dei vari nodi per ampliare la conoscenza dell'ambiente in cui il veicolo si trova in ogni dato istante.

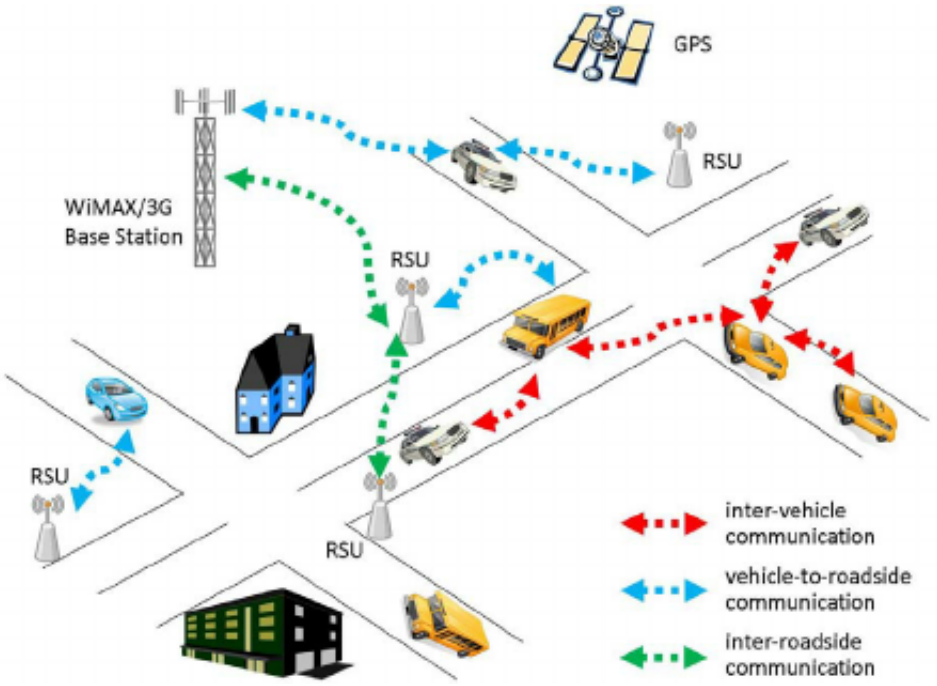


Figura 2.1: Scenario di esempio di una rete VANET (fonte: [1])

2.1 Componenti

Come già anticipato nel precedente paragrafo, nell'ambito delle VANET vengono presentati una serie di dispositivi hardware ognuno avente un compito specifico con l'obiettivo di mettere in piedi una rete di comunicazione solida, robusta, efficiente e sicura [2].

Segue una breve panoramica di ogni singolo componente partendo dalla RSU.

2.1.1 RoadSide Unit (RSU)

La Roadside Unit è un dispositivo fisso posto in genere lungo il bordo dell'asse stradale o in luoghi con alta concentrazione di traffico, ad esempio un'area di servizio, connesso all'infrastruttura di rete Internet esistente attraverso una scheda apposita ed equipaggiato con una ulteriore scheda di rete capace di implementare il protocollo *IEEE 802.11p*. I principali compiti di una RSU sono:

- mantenere attiva una rete di comunicazione con le altre RSU al fine di scambiare le informazioni imparate nelle varie parti della rete;
- comunicare le informazioni recepite ai nodi mobili (veicoli) attraverso le OBU mettendo in atto una comunicazione V2I (*Vehicle-to-Infrastructure*);
- offrire la connettività a Internet alle OBU.

2.1.2 OnBoard Unit (OBU)

La OBU è un dispositivo hardware installato a bordo di un veicolo costituito da una serie di componenti quali GPS (*Global Positioning System*), memoria EPROM, scheda di rete e un piccolo processore. Questo componente funge da trasmettitore e ricevitore. Il principale compito è quello di scambiare informazioni con altre OBU e con le RSU incontrate lungo il tragitto seguito dal veicolo attraverso un canale wireless basato sullo standard IEEE 802.11p. Talvolta può essere dotato di un'opportuna scheda di rete per comunicare in GSM, UMTS oppure 4G, con altri host sulla rete Internet al fine di fornire dei servizi aggiuntivi come la trasmissione di informazioni relative alla situazione del traffico in tempo reale ai veicoli durante i loro spostamenti.

2.1.3 Application Unit (AU)

La Application Unit è un componente con capacità di processing delle informazioni posto direttamente a bordo del veicolo che utilizza ed esegue le applicazioni fornite dal Provider e che sfrutta le capacità di comunicazione con la rete fornite dalla OBU alla quale può connettersi mediante una connessione wired o wireless. Talvolta la AU può fondersi con la OBU in un unico dispositivo fisico da installare sul veicolo.

2.1.4 Altri componenti

Oltre ai dispositivi appena presentati ogni veicolo possiede altri tre componenti fondamentali:

- **TPD (Tamper Proof Device) & TPM (Trusted Platform Module)** sono due dispositivi orientati alla sicurezza. Il TPD è un componente completo, sia software che hardware, che svolge due funzioni principali:
  - *generazione e memorizzazione dei dati utili per la cifratura*;
  - *operazioni crittografiche:*
    essendo un device con particolari capacità di processing viene spesso utilizzato per eseguire operazioni di cifratura sui dati da trasmettere agli altri nodi e di verifica su quelli ricevuti.

Il TPM è un modulo hardware che può all'occorrenza prendere il posto del TPD in quanto assolve alle stesse funzioni anche se il suo costo è molto inferiore nonostante comprenda diversi componenti funzionali aventi capacità crittografiche (RSA engine, SHA-1 engine e generatore di numeri casuali). A differenza del primo, il TPM necessita di un software apposito per potervi comunicare e, al fine di proteggere l'integrità del sistema da eventuali manipolazioni, nella fase di boot calcola e salva l'impronta di ogni componente sul PCR (*Platform Configuration Register*) permettendogli in ogni istante futuro di valutarne lo stato ed individuare eventuali discrepanze dovute a possibili manomissioni. Esso possiede una EK (*Endorsement Key*) corrispondente ad una coppia di chiavi RSA avente una dimensione minima di 2048 bit generata e salvata sul modulo direttamente dal costruttore ed inoltre ha la capacità di autogenerarsi una serie di chiavi *AiK* per il cosiddetto *Attestation Protocol* che gli consentirà in ogni momento di verificare i dati ricevuti senza mai esternare l'identità del TPM . Questi dispositivi sono in genere "tamper-resistant" ovvero possono resistere ad eventuali manomissioni fisiche. I loro limiti principali sono il costo e la temperatura d'esercizio che non deve essere troppo elevata per non causare dei malfunzionamenti.

- **ELP (Electronic License Plate)** è un componente che contiene l'identità digitale del veicolo rilasciata da una entità specifica autorizzata.
- **EDR (Event Data Recorder)** è un dispositivo simile alla scatola nera presente sugli aerei che registra gli eventi e i dati critici dei veicoli durante situazioni di emergenza.

2.2 Comunicazione e applicazioni di rete

Un messaggio può "appartenere" ad una specifica applicazione la quale può essere idealmente associata ad una delle seguenti quattro classi:

- Pubblica Sicurezza (situazioni di emergenza);
- Ausilio alla Guida (informazioni sul traffico o sul meteo);
- Sicurezza Attiva (incidenti o situazioni di pericolo sulla carreggiata);
- Business/Entertainment.

Due possibili esempi di applicazioni di sicurezza sono il *Forward Collision Warning* (FCW) e l'*Emergency Electronic Brake Lights* (EEBL) (Tabella 2.1).

Una certa applicazione può fare affidamento a due tipi di messaggi: messaggi generati periodicamente contenenti informazioni di uno specifico tipo (*application messages*) oppure messaggi generati in seguito al verificarsi di un evento (*event-based messages*). I messaggi per la sicurezza stradale, che da adesso in poi verranno

Applicazione	**Obiettivo**
Intersection Collision Warning (ICW)	Limitare il rishio di incidenti agli incroci stradali
Lane Change Assistance (LCA)	Evitare possibili situazioni di pericolo in seguito a cambi di corsia da parte di veicoli che precedono
Rear End Collision Warning (RECW)	Limitare i tamponenti con veicoli che precedono in seguito a incidenti o bruschi rallentamenti
Emergency Electronic Brake Light (EEBL)	Informare i veicoli che sopraggiungono di situazioni di pericolo derivanti da brusche frenate di emergenza
Stationary Vehicle Warning (SVW)	Comunicare la presenza di un veicolo per qualche ragione (incidente o problema meccanico) fermo nella carreggiata
Traffic Condition Warning (TCW)	Comunicare agli altri nodi della rete (veicoli ed RSU) una situazione di traffico in evoluzione
Hazardous Location Notification (HLN)	Segnalare possibili situazioni di pericolo dovute alla presenza di ostacoli sulla carreggiata o di cantieri stradali

Tabella 2.1: Principali applicazioni di sicurezza attiva

chiamati anche *beacon*, contengono una serie di informazioni come la posizione geografica del veicolo, la direzione e la velocità del veicolo o ancora l'identificativo del veicolo che li hanno generati.

In realtà, attraverso numerosi studi ed esperimenti, si è cercato di capire quale sia il metodo più efficiente da adottare per la comunicazione inter-veicolare e la comunità scientifica ha ampiamente discusso sulla possibilità di impiego di una strategia (algoritmo) *beacon-less* o *beaconed* [3]. In quest'ultima i beacon sono fortemente utilizzati da un nodo, sia nel caso di comunicazione unicast che broadcast, per effettuare un'esplorazione dell'area in cui si trova e quindi per conoscere il vicinato col quale effettuare poi lo scambio di dati. La prima strategia invece si basa sulle informazioni in tempo reale relative alla posizione dei nodi e comporta un overhead maggiore rispetto a quella basata su beacon ma di contro dà maggiori garanzie nella trasmissione dei dati offrendo alta affidabilità e tolleranza agli errori del canale di comunicazione. Data la sua importanza nel contesto VANET, il processo di beaconing verrà ripreso e approfondito nel Capitolo 3.

Uno dei principali campi di ricerca è quello relativo alla propagazione del segnale attraverso il canale wireless che per sua natura è dinamico e a causa della sua variabilità, legata anche alla penetrabilità dell'ambiante nel quale viene inviato, si potrebbero talvolta presentare dei gravi problemi durante lo scambio di informazioni. Al fine di studiare correttamente attraverso le simulazioni ogni possibile scenario reale di una rete VANET sono stati sviluppati alcuni modelli di trasmissione e propagazione del segnale come il *Free Space Model*, il *Nakagami Model* ed il *Log Normal Shadowing Model*. Volendo fare un esempio è possibile, sfruttando l'*Equazione di Friis* (equazione 2.1), calcolare il range di trasmissione del segnale nell'aria nel caso del **Free Space Model** [4].

$$P_r = \frac{G_t G_r \lambda^2}{(4\pi R)^2} P_t \tag{2.1}$$

Nella formula P_t e P_r fanno riferimento rispettivamente alle potenze di trasmissione e ricezione del segnale, G_r e G_t sono i guadagni del segnale di ricezione e trasmissione delle antenne, λ è la lunghezza d'onda ed R è la distanza tra i nodi.

Unitamente alle due strategie prima citate (beacon-si/beacon-no) un ampio numero di sperimentazioni e ricerche sono state condotte per la definizione del miglior protocollo di routing da adottare nelle VANET.

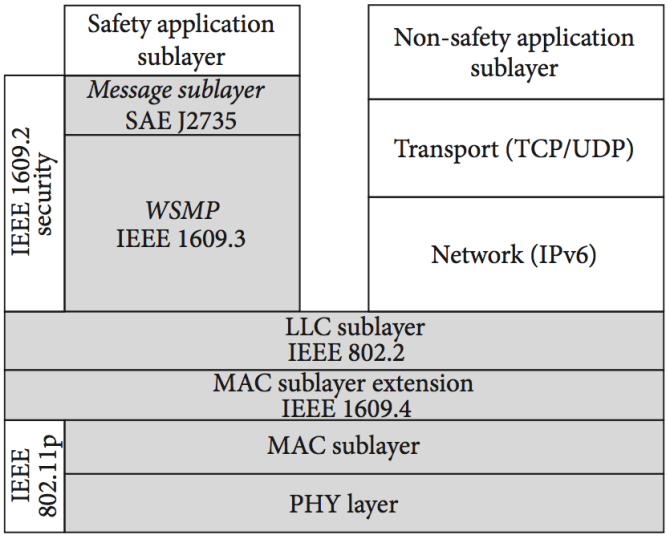


Figura 2.2: Stack di rete e relativi standard

2.3 Lo standard IEEE 802.11p

IEEE 802.11p è lo standard de facto per le Vehicular Ad-Hoc Network e deriva dal più famoso IEEE 802.11 che è stato inizialmente pensato per uno scenario differente in cui i nodi di rete non avessero la stessa dinamicità e velocità di spostamento dei veicoli. In linee generali, esso risulta particolarmente adatto a quegli ambienti in cui

le proprietà del livello fisico cambiano rapidamente, proprio come nella tipologia di rete qua trattata e, rispetto all'802.11, aggiunge alcune migliorie per il supporto alle applicazioni per l'Intelligent Transportation System. Una delle aggiunte riguarda ad esempio la possibilità di scambio di messaggi tra veicoli ad alte velocità e tra veicoli ed RSU.

Questo standard è utilizzato nei primi due livelli dello stack protocollare DSRC (livello fisico e livello di accesso al mezzo) (Figura 2.2) e viene indicato dalla sigla WAVE che sta per *Wireless Access in Vehicular Environment*. Per i livelli successivi vengono invece utilizzati gli standard IEEE 1609.x :

- IEEE 1609.4 è relativo allo switching del canale;
- IEEE 1609.3 fa riferimento ai servizi di rete;
- IEEE 1609.2 è per i servizi di sicurezza.

Come indicato in [2] e mostrato dalla Figura 2.3, la tecnologia DSRC (*Dedicated Short-Range Communication*) sfrutta ben 75 MHz di banda nello spettro di frequenza dei 5.9 GHz (5.850 - 5.925) e lo divide in 7 canali ognuno da 10MHz circa utilizzabili da una vasta gamma di applicazioni. WAVE è in grado di gestire un ambiente nel quale i nodi mobili possono spostarsi ad una velocità massima di oltre 260 km/h coprendo una distanza compresa tra 300 e 1000 metri con un data rate fino a 27Mbps. I canali dedicati alle applicazioni per la sicurezza sono il 172 (5.855 - 5.865) ed il 184 (5.915 - 5.925) mentre 4 sono i canali bidirezionali dedicati alla comunicazione tra le diverse entità (174, 176, 180 e 182) che in realtà possono anche essere combinati a coppie per dar vita a due soli canali (175 e 181) con un'ampiezza di banda di 20MHz piuttosto che da 10MHz. Il senso dell'utilizzo del canale da 10 MHz, invece che da 20 MHz, è connesso all'esigenza di utilizzare più applicazioni in parallelo e quindi di ridurre la congestione dei canali di comunicazione a disposizione. Vari test fisici condotti in questo ambito hanno dimostrato che la scelta di canali da 10 MHz risulta più efficiente sia per i ritardi che per l'effetto doppler tipico degli ambienti veicolari.

Dal punto di vista pratico possono essere adoperati diversi schemi di modulazione del segnale per una efficiente trasmissione dei pacchetti. Tipicamente si fa affidamento alla modulazione OFDM (*Orthogonal Frequency Division Multiplexing*) che viene in genere usata per lo standard IEEE 802.11a con l'apporto di alcune modifiche ai parametri come la maschera di trasmissione o la potenza irradiata.

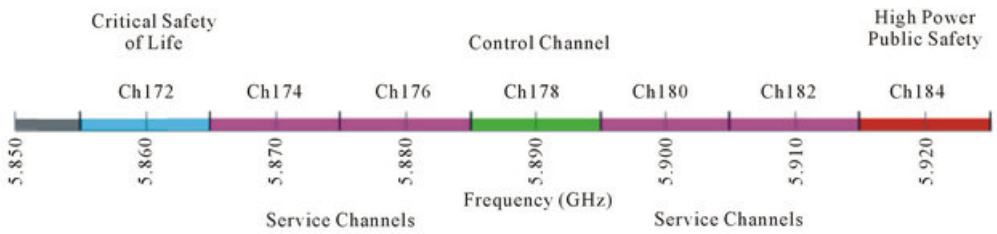


Figura 2.3: Canali di comunicazione WAVE

Addentrandoci nelle specifiche dello standard, diamo un leggero sguardo ai livelli PHY e MAC. Il livello fisico (PHY), presentato e discusso minuziosamente in [5], è composto da due sottolivelli:

- **PLCP (Physical Layer Convergence Protocol)** è responsabile per la comunicazione con il livello superiore (MAC);
- **PMD (Physical Medium Access)** funge da interfaccia con il mezzo fisico utilizzato.

Lo strato MAC è basato sul CSMA/CA (*Carrier Sense Multiple Access / Collision Avoidance*) usato anche dall'802.11, dal quale però si differenzia in quanto non sono effettuate associazioni e autenticazioni per ovvie motivazioni connesse al rendimento della rete.

Il *WAVE mode* mette a disposizione due distinti canali di comunicazione: SCH (*Service Channel*) e CCH (*Control Channel*). Il SCH è utilizzato per le comunicazioni WBSS (*Wave Basic Service Set*) tra Provider e User, ovvero tra colui che offre i servizi e gli utenti finali che ne fanno uso. Il CCH è, tra i due canali, il più importante. Su di esso tutte le stazioni si mettono in ascolto per poter recepire eventuali messaggi di sicurezza o annunci WBSS inviati in broadcast sulla rete. Per questo motivo si dice che questo è tra tutti l'unico canale condiviso da tutti i nodi della rete.

Una WBSS è un tipo di BSS che consiste in una serie di veicoli/stazioni che cooperano in modalità WAVE attraverso una comunicazione basata sullo stesso indirizzo BSSID il cui tempo di vita è strettamente connesso alla presenza di membri al suo interno. Per le applicazioni di sicurezza esiste un indirizzo speciale *wildcard BSSID* mediante il quale un veicolo facente parte di una certa stazione WBSS può comunicare una situazione di pericolo a tutti i veicoli vicini indipendentemente dalla WBSS della quale fanno parte attraverso l'invio di un beacon.

Il sottolivello MAC corrisponde all'IEEE 802.11e EDCA (*Enhanced Distributed Channel Access*) che mette a disposizione il servizio di TXOP (*Traffic Opportunity*) secondo il quale vengono definite 4 code di servizio dette CFA (*Channel Access Function*) differenziate da parametri diversi (da AC0 priorità minima fino a AC3 priorità massima) per la contesa di accesso al canale. Queste code definiscono 4 classi di servizio per una gestione ottimale della QoS e sono chiamate rispettivamente Background Traffic (BK o ACO), Best Effort Traffic (BE o AC1), Video Traffic (VI o AC2) e Voice Traffic (VO o AC3). In questo modo sarà possibile assegnare ai messaggi delle priorità in funzione della loro "importanza" al fine di limitare eventuali danni causati dai possibili ritardi di cui questa tipologia di rete può soffrire.

Oltre all'IEEE 802.11p e all'IEEE 1609 è stato definito un terzo standard chiamato J2735 da parte della SAE (*Society of Automotive Engineers*) che specifica la struttura e la tipologia dei messaggi che possono essere generati dalle varie applicazioni [6]. L'insieme dei messaggi specificato dallo standard J2735 definisce il contenuto del messaggio consegnato dal sistema di comunicazione (OBU) al livello applicazione e quindi definisce il payload del messaggio al livello fisico. Tra le varie tipologie di messaggi è possibile trovare: messaggi per la sicurezza stradale, messaggi generici per le varie applicazioni e messaggi di allerta per i casi di emergenza.

2.4 QoS nelle reti VANET

Il concetto di Qualità del Servizio (QoS) riferito ad una generica rete non ha una definizione univoca ma in genere sta ad indicare la misura della qualità della comunicazione e di accessibilità ad un certo servizio offerto all'utente finale dalla stessa rete. In effetti, nel caso delle reti infrastrutturate come la rete Internet, un ISP (Internet Service Provider) potrebbe offrire una vasta gamma di servizi ai propri clienti che a loro volta potrebbero avere una diversa percezione della qualità del servizio offertagli.

Gli algoritmi per la realizzazione della QoS operano in diversi strati della pila protocollare. I principali criteri che consentono di valutare la qualità del servizio offerto da una rete possono essere:

- larghezza di banda minima;
- tasso di perdita dei pacchetti;
- ritardo massimo di trasmissione;
- jitter.

Nel caso specifico delle VANET, il traffico di rete può essere classificato come *real-time* o *non-real-time* e richiede requisiti differenti per la qualità del servizio. La corretta trasmissione dei pacchetti in un intervallo di tempo ristretto e fissato è una criticità di non poco conto. Le applicazioni di Sicurezza Stradale richiedono una bassissima latenza di trasmissione end-to-end perché se i messaggi dovessero arrivare a destinazione con un grosso ritardo potrebbero di fatto risultare inservibili ai fini della sicurezza degli utenti.

Da quanto appena detto, è facile intuire che la definizione della QoS risulta particolarmente difficile a causa della dinamicità topologica di questi ambienti e di conseguenza si preferisce fare affidamento ad una serie di metriche che dipendono dai vari livelli dello stack di rete per la valutazione della qualità e dell'integrità delle comunicazioni.

La Tabella 2.2, che riassume e descrive le suddette metriche, non fa alcun riferimento ad un "parametro" di particolare rilevanza: la sicurezza nella comunicazione. Questo concetto potrebbe sembrare separato da quello di QoS ma in realtà non lo è e, data la sua importanza, ci riserviamo di trattarlo successivamente (Capitolo 4) con maggiore enfasi rispetto a quanto fatto per gli altri nel presente paragrafo.

Diverse soluzioni sono state proposte per la QoS nelle reti MANET anche se bisogna dire che il loro porting nell'ambito VANET è risultato piuttosto difficoltoso. Tra questi troviamo le due interessanti classi di protocolli di routing reattivi e proattivi che saranno discussi nel paragrafo successivo.

In [7] viene proposto un nuovo protocollo di livello MAC chiamato MQOG (*Multichannel QoS Cognitive MAC*) che include efficienti tecniche per la negoziazione del canale e per la valutazione del canale di trasmissione al fine di migliorare l'affidabilità della comunicazione ed il throughput. Esso fa uso di una tabella CNST

(*Channel Neighbor State Table*) per tenere traccia di tutte le comunicazioni attive con i veicoli vicini istante per istante con aggiornamenti continui. Inoltre viene messo in atto un particolare meccanismo che permette una gestione prioritaria di quei messaggi appartenenti a tutte quelle applicazioni orientate alla sicurezza che richiedono bassi ritardi di trasmissione.

Parametro	**Proprietà**
Larghezza di banda	Indica l'ampiezza della banda disponibile
Perdita dei pacchetti	Indica il tasso o la quantità dei pacchetti andati persi per cause diverse come la congestione della rete
Velocità	Indica quanto rapidamente un pacchetto fluisce sulla rete
Comunicazione end-to-end	Indica lo stato e l'integrità del canale di comunicazione utilizzato dai messaggi per raggiungere la destinazione
Error rate	Indica il tasso di errore che può presentarsi durante la comunicazione
Jitter	Indica la qualità della comunicazione attraverso la valutazione della varianza del ritardo di trasmissione dei pacchetti sulla rete

Tabella 2.2: Parametri per la valutazione della QoS nelle reti VANET

2.5 Il routing nelle VANET

Il principale obiettivo di qualsiasi algoritmo di routing è quello di trovare una via di comunicazione ottimale tra i nodi di una rete. In particolare, una caratteristica fondamentale ricercata in un generico protocollo di routing per la gestione delle rotte su una rete VANET è senza dubbio la velocità di trasmissione, ovvero la rapidità con cui i messaggi possono fluire sul canale di comunicazione fino a raggiungere il destinatario o i destinatari, unita al minor consumo di risorse di rete possibile.

Inizialmente si pensava che fosse possibile adattare in modo veloce e senza problemi di sorta i protocolli già presenti e sviluppati per le MANET ma ci si è poi resi conto che a causa delle sostanziali differenze, tanto per citarne una la dinamicità e la velocità di movimento dei nodi all'interno della rete e quindi la conseguente rapidità di mutazione della topologia della stessa, sarebbe stato preferibile iniziare un nuovo studio ed uno sviluppo ad-hoc per le VANET.

La definizione di una rotta per l'instradamento dei pacchetti non è un'operazione immediata come può esserlo per il caso delle reti aventi un'infrastruttura ben definita poiché nelle VANET i vari hop sono costituiti da nodi che variano continuamente la loro posizione sulla rete. Per questa ragione il sistema molto spesso adottato per la creazione delle rotte è il broadcast e in particolare, come avremo modo di vedere, il flooding. Seguendo questa strategia, un nodo che riceve un messaggio si preoccuperà semplicemente di ritrasmetterlo ai propri vicini una sola volta. Purtroppo, come è facile capire, questa tecnica causa un drastico aumento del numero di pacchetti ridondanti sulla rete in seguito alle continue ritrasmissioni da parte dei nodi e quindi una diminuzione delle prestazioni della rete stessa. Al fine di ridurre queste problematiche sono state proposte diverse soluzioni in letteratura come [8, 9] che in generale si preoccupano di limitare i danni causati da questi algoritmi mediante delle ottimizzazioni che cercano di ridurre il numero di ritrasmissioni dei pacchetti e quindi di messaggi duplicati sulla rete.

In [10] vengono discusse diverse strategie organizzative ed in special modo viene posta particolare attenzione sulle proprietà dei diversi protocolli di routing presenti in tale ambito, i quali vengono poi suddivisi logicamente in cinque diverse famiglie: ad-hoc o topology driven routing, location-based routing, cluster-based routing, broadcast routing e geocast routing. Partendo dalla prima, facciamo un piccolo excursus sulle proprietà delle diverse famiglie di protocolli.

2.5.1 Protocolli di routing topology driven

Questa famiglia di protocolli viene suddivisa a sua volta in tre sottocategorie: protocolli proattivi, reattivi e ibridi. In un protocollo di tipo proattivo i nodi mantengono in memoria ed aggiornano regolarmente le proprie tabelle di routing attraverso le informazioni relative alle rotte che vengono scambiate all'interno della rete. Queste informazioni vengono inviate periodicamente dai nodi mediante lo scambio di pacchetti di HELLO i quali però non solo causano un sovraccarico alla rete ma consumano parte della larghezza di banda disponibile. Naturalmente la crescita del numero di nodi sulla rete comporta un vertiginoso aumento della dimensione della tabella di routing che ogni nodo deve "gestire". Questo è il motivo principale per cui questo tipo di protocolli non sono particolarmente preferiti per reti di grosse dimensioni. Al contrario, la seconda categoria di protocolli, quelli di tipo reattivo, si limitano soltanto allo scambio ed invio di informazioni solo ed effettivamente quando ce n'è di bisogno, evitando così questo flusso continuo di pacchetti sulla rete. In questo caso l'overhead è drasticamente ridotto ed è semplicemente legato alla sola ricerca del percorso sul quale inviare le informazioni di rete. Per questa ragione questa tipologia di protocolli è da preferire nel caso di reti la cui conformazione topologica varia in continuazione. Il processo di ricerca del percorso tra due nodi sulla rete è eseguito sfruttando dei messaggi speciali chiamati RREQ (*Route Request Message*).

In generale è possibile affermare che il ritardo end-to-end è più elevato nei protocolli di tipo reattivo rispetto a quelli di tipo proattivo. Un sommario con le principali proprietà e differenze di queste due famiglie di protocolli topology driven è definito nella Tabella 2.3.

Infine, i protocolli di tipo ibrido sono caratterizzati da un mix delle principali proprietà dei protocolli di tipo reattivo e proattivo. In particolare la rete viene divisa in tante zone al fine di evitare il principale problema dei protocolli di tipo proattivo e quindi di ridurre il tempo connesso al processo di discovery delle rotte sulla rete. Precisamente la filosofia di funzionamento dei protocolli reattivi viene usata per la comunicazione tra le diverse zone mentre quella dei protocolli proattivi la si utilizza all'interno delle zone stesse.

Proprietà	**Protocollo reattivo**	**Protocollo proattivo**
Disponibilità di informazioni di routing	Quando necessaria	Sempre disponibili
Distribuzione della topologia	Su richiesta	Periodica
Aggiornamento periodico delle rotte	Non richiesto	Richiesto
Informazioni di routing	Non memorizzate	Memorizzate nelle tabelle di routing
Ritardo di trasmissione	Alto	Basso
Controllo del traffico	Basso	Alto

Tabella 2.3: Confronto proprietà principali protocolli di tipo reattivo e proattivo

2.5.2 Protocolli di routing location based

I protocolli appartenenti a questa famiglia utilizzano fortemente le informazioni legate alla posizione geografica di ogni nodo ottenibili mediante sistemi appositi come il GPS (*Global Positioning System*) e a differenza di altri non devono preoccuparsi di gestire e aggiornare le rotte sulla rete. Nel momento in cui un nodo decidesse di inviare un pacchetto ad un certo altro nodo destinatario basterà inserire l'informazione sulla posizione del destinatario nell'header del pacchetto stesso. Quindi il processo di invio di un pacchetto si limita alla semplice ricerca della posizione geografica del destinatario ed al successivo inoltro mediante un opportuno servizio.

Questa tipologia di protocolli è senza dubbio da preferire per ambienti dinamici come le reti VANET. Tuttavia, come in ogni cosa, ci sono delle limitazioni. La presenza del dispositivo GPS è di vitale importanza. Se questo, per qualsiasi

ragione, non dovesse funzionare a dovere il protocollo potrebbe non svolgere il suo lavoro secondo la sua filosofia costruttiva. Inoltre questi protocolli possono soffrire di problemi alquanto gravi quali routing loop e ritardi nella trasmissione dei pacchetti.

2.5.3 Protocolli di routing cluster-based

Al fine di ridurre il carico ed il traffico sulla rete si è pensato di creare un'architettura di rete basata su piccoli gruppi di veicoli chiamati *cluster*. La scelta dei componenti di un cluster può essere legata ad alcune caratteristiche dei nodi come la velocità o la direzione. All'interno di ogni gruppo vi è un nodo che funge da "capo" che è anche il responsabile della comunicazione tra i diversi cluster presenti nella rete.

La composizione di un cluster può essere definita da un numero che specifica la quantità massima di veicoli presenti al suo interno oppure semplicemente dall'insieme dei veicoli che si trovano geograficamente in una certa area. L'invio di messaggi tra nodi all'interno di un cluster avviene attraverso delle rotte "dirette" mentre la comunicazione tra i diversi gruppi avviene attraverso la creazione di una infrastruttura di rete virtuale. Naturalmente al crescere del numero di nodi all'interno dei cluster o del numero di cluster nella rete, aumenta anche il carico complessivo che può comportare dei ritardi non indifferenti nella trasmissione dei pacchetti sulla rete.

2.5.4 Protocolli di routing broadcast

La modalità di trasmissione delle informazioni sulla rete usata da questi protocolli è il *broadcast* e in particolar modo il *flooding*. Tuttavia, se da un lato questo assicura che i pacchetti arrivino a destinazione, dall'altro apporta un grosso sovraccarico a causa della larghezza di banda richiesta in quanto una moltitudine di pacchetti duplicati saranno ad un certo istante presenti sulla rete. Questa tipologia di protocolli è ampiamente utilizzata per le applicazioni di sicurezza le quali richiedono che le informazioni giungano repentinamente e con assoluta certezza a destinazione.

2.5.5 Protocolli di routing Geocast

Un protocollo di routing appartenente a questa famiglia fonda il suo funzionamento sull'idea di disseminare, attraverso una comunicazione multicast, delle informazioni all'interno di una certa area della rete. Basti pensare al caso di un incidente sull'asse stradale. Con questa tipologia di protocolli è possibile trasmettere questa informazione a tutti i nodi che si trovano nelle immediate vicinanze e che quindi sono fortemente interessate all'accaduto. L'area interessata all'invio dei messaggi è indicata con la sigla ZOR (*Zone Of Relevance*). Un certo pacchetto potrebbe poi essere anche inoltrato ai nodi delle aree limitrofe che fanno parte della cosiddetta ZOF (*Zone Of Forwarding*). Il principale problema di questa famiglia di protocolli è il ritardo di trasmissione dei pacchetti causato dalle continue disconnessioni dei nodi presenti sulla rete.

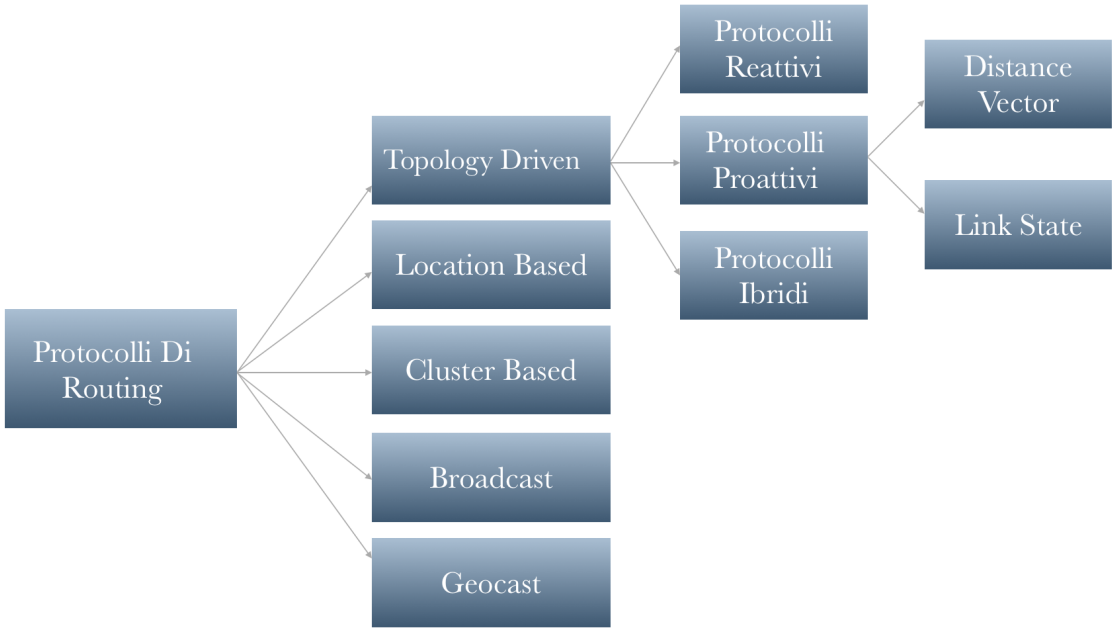


Figura 2.4: Classificazione completa dei protocolli di routing

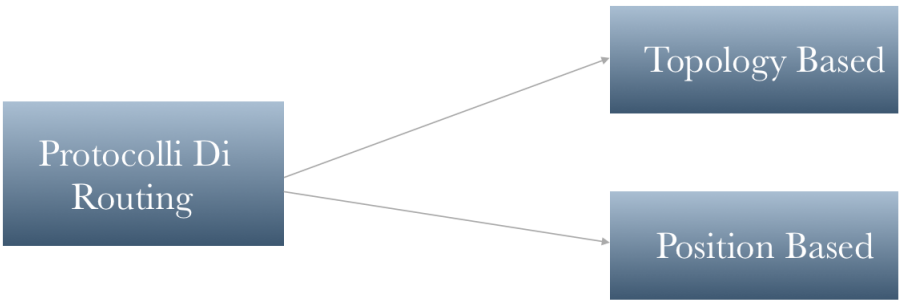


Figura 2.5: Classificazione (alternativa) compatta dei protocolli di routing

2.5.6 Considerazioni finali sul routing

Nonostante la diversità di categorie e protocolli presenti, spesso si tende a raggruppare in modo molto più generico i protocolli di routing in due sole classi: *topology-based* e *position-based*. I primi sfruttano le informazioni sullo stato dei link all'interno della rete al fine di consentire una corretta trasmissione dei pacchetti dalla sorgente alla destinazione. Fanno parte di questa classe protocolli come FSR e AODV. I secondi invece sfruttano soltanto le informazioni sulla posizione al fine di prendere le decisioni per il corretto instradamento dei pacchetti sulla rete. Questi ultimi, a differenza dei protocolli topology-based, non gestiscono alcuna tabella di routing e non si basano sulle informazioni relative allo stato dei collegamenti sulla rete ma fondano la loro filosofia di funzionamento sul fatto che ogni nodo conosce se stesso,

o meglio la propria posizione sulla rete, e tutti i nodi che si trovano nel proprio vicinato. Di questa seconda famiglia fanno parte protocolli come DTN e GPSR.

L'instradamento geografico basato sulla posizione dei veicoli necessita di un meccanismo capace di risolvere la posizione del nodo destinatario a partire dalla sua identità. Se nella Internet è il protocollo ARP a preoccuparsi mediante un processo query-reply basato sul broadcast della risoluzione di un indirizzo fisico, nelle VANET è presente il cosiddetto *Location Service* che svolge un ruolo simile. Nel Paragrafo 2.7.1 verrà ripreso e discusso con più attenzione il ruolo e la possibile implementazione di questo importantissimo servizio.

Ritornando al problema del routing, la Tabella 2.4 riassume alcuni protocolli proposti nel corso degli anni che hanno suscitato maggiore interesse da parte della comunità scientifica mentre [11] li mette a confronto valutandone pro/contro e relative prestazioni.

2.6 Evoluzione del routing

A causa dell'elevata dinamicità delle reti VANET si può avere, a seconda della densità di veicoli, una abbondanza o al contrario una scarsità di messaggi che vengono immessi sulla rete stessa e quindi di nodi che possono poi provvedere alla loro successiva ritrasmissione. Questa problematica incoraggia, ad esempio, lo sviluppo di nuove tecniche di buffering dei pacchetti al fine di consentire una corretta propagazione delle informazioni verso tutti i nodi interessati. Unitamente a quanto appena detto, diverse ricerche sono state condotte relativamente allo studio di nuovi protocolli di routing al fine di garantire una maggiore affidabilità nella propagazione dei messaggi sulla rete. L'obiettivo principale è stato quello di provare a mettere in piedi un algoritmo adattivo tale per cui il suo "comportamento" possa adattarsi ai continui cambiamenti di stato di una rete VANET. Una delle soluzioni è la cosiddetta architettura di rete *component-based* che, facendo uso di componenti riutilizzabili, propone la possibilità di variare automaticamente i parametri di rete in funzione dello stato della rete stessa. In [14] viene studiata e proposta una possibile implementazione di questo tipo e vengono anche messe in evidenza le possibili similitudini tra i principali protocolli di routing presenti ad oggi al fine di costruire una piattaforma generica di tipo component-based da usare per la definizione di nuovi protocolli.

Oltre a quest'ultima strategia sono stati sviluppati dei protocolli particolarmente adatti alle reti sparse che prendono il nome di *protocolli delay-tolerant*. Nello specifico caso delle VANET le reti sparse si possono ad esempio trovare nelle zone rurali di campagna dove il numero di veicoli e quindi di nodi specie nelle ore diurne è di gran lunga inferiore rispetto alle zone cittadine nelle quali è possibile invece trovare le cosiddette reti dense. Una rappresentazione delle due tipologie di reti, sempre nel campo VANET, è mostrata dalla Figura 2.6 dove, tramite uno scenario relativo ad un incidente stradale, è possibile distinguerne la natura. In questi casi risulta praticamente impossibile definire una rotta tra due end-point e perciò si adotta la tecnica di trasmissione *carry-and-forward* (letteralmente trasporta-e-inoltra), con la

Protocollo	**Tipologia**	**Overhead**
AODV	Topology-Based	Path States
FSR	Topology-Based	Link States
AODV + PGB	Topology-Based	Path States
DSR	Topology-Based	Path States
TORA	Topology-Based	Path States
GPSR	Position-Based	Beacons
GRANT	Position-Based	Two-Hop Beacons
GPSR + AGF	Position-Based	Beacons
CAR	Position-Based	Beacons + Path States
GSR	Position-Based	Beacons
CBF	Position-Based	Data Broadcast
TO-GO	Position-Based	Beacons + Data Broadcast
Ge-Opps	Position-Based	Beacons
VADD	Position-Based	Beacons
GeoDTN + NAV	Position-Based	Beacons

Tabella 2.4: Sommario dei principali protocolli di routing per le VANET [12]

quale un pacchetto è inoltrato solo ed esclusivamente quando ci sono nodi disponibili altrimenti viene semplicemente memorizzato e trasportato dal veicolo stesso che lo detiene fino al suo successivo inoltro. In [15] viene mostrato come una corretta cooperazione tra i nodi mobili ed il meccanismo carry-and-forward, unitamente ad una gestione mediante priorità delle due tecniche di inoltro dei pacchetti, possono effettivamente migliorare le prestazioni delle reti sparse per la trasmissione dei dati.

Questi appena citati sono solo un paio di esempi che abbiamo preso in considerazione per mettere in risalto l'importanza che ricopre un protocollo di routing per una rete VANET, nel nostro specifico caso, e gli sforzi che tutt'oggi continua a fare l'intera comunità scientifica per la ricerca e la definizione di una tecnica ottimale per la trasmissione dei messaggi tra i nodi che compongono la suddetta infrastruttura

di rete.

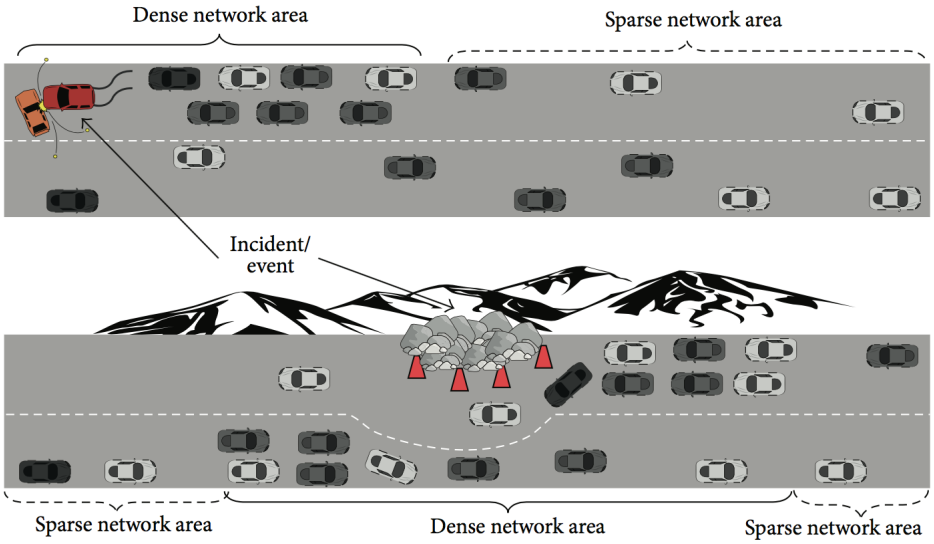


Figura 2.6: Confronto grafico tra rete sparsa e rete densa (fonte: [13])

2.7 Come conoscere la posizione dei nodi per l'invio dei messaggi ?

In uno dei precedenti paragrafi è stato messo in evidenza un componente di vitale importanza soprattutto per quei protocolli basati sul cosiddetto routing geografico: il *Location Service* (LS). Come prerequisito fondamentale di un protocollo di routing position-based, ogni nodo della rete deve conoscere la posizione del nodo con il quale vuol comunicare.

Lo sviluppo di un LS per le VANET è senza dubbio un compito non facile. Il principale ostacolo è connesso al rendimento di questo servizio poiché anche da esso dipendono le prestazioni dell'intera rete. Una corretta scelta del location server può portare ad un aumento del numero di pacchetti consegnati nell'unità di tempo grazie alla minimizzazione del ritardo per la "risoluzione della posizione" del destinatario. Questo processo richiede un certo grado di sicurezza e affidabilità in quanto potrebbe accadere che un nodo fittizio possa spacciarsi per un location server autorizzato e disseminare informazioni appositamente manipolate sulla rete. L'architettura costruttiva del LS si basa su due processi principali. Il primo, chiamato *Location Update* (LU), si preoccupa della gestione delle informazioni della posizione dei vari nodi: ogni nodo invia degli update relativi alla propria posizione ad uno o più server utilizzati da questo processo per comunicargli costantemente le informazioni aggiornate sulla propria posizione. Il secondo, detto *Location Request* (LR), è il gestore delle richieste per la risoluzione della posizione dei nodi: un nodo, che chiamiamo sorgente, può inviare una richiesta per avere informazioni relative alla posizione di un certo altro nodo destinatario al quale vuol inviare dei dati.

In [16] viene proposta e valutata un'architettura ibrida centralizzata per il Location Service che fa uso dell'infrastruttura di rete esistente (WAN+4G). Si tratta di uno studio che, per certi versi, può essere definito anomalo ma allo stesso tempo interessante. Questa affermazione si basa sul fatto che la stragrande maggioranza delle ricerche condotte in questo ambito basano le loro teorie su una rete che non ha un'infrastruttura specifica, cioè una rete in cui sono gli stessi nodi mobili che svolgono il compito di Location Service.

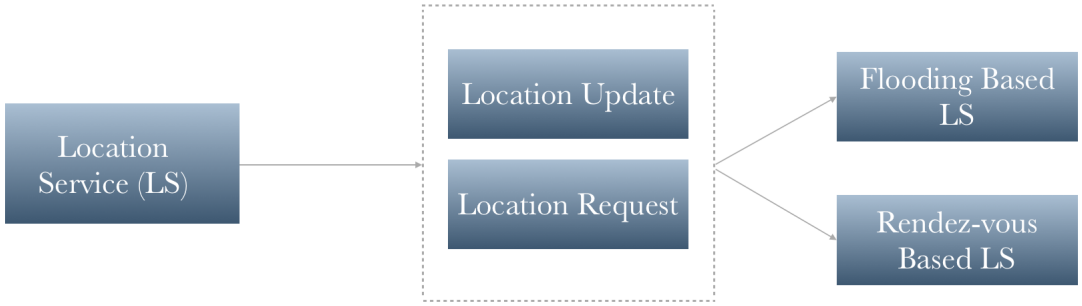


Figura 2.7: Struttura e composizione del Location Service

2.7.1 Strategie per l'implementazione del LS

Un LS può essere implementato seguendo due diverse strategie. Nella prima, denominata *flooding-based LS*, ogni nodo della rete funge sia da location server che da client. In sostanza, ogni veicolo si preoccupa di inviare periodicamente in flooding le informazioni che ha imparato e di inoltrare delle richieste ad altri server per conoscere le posizioni di altri veicoli, quando lo ritiene opportuno. Nel secondo approccio *rendez-vous based LS* alcuni veicoli svolgono il ruolo specifico di location server e contengono le informazioni sulla posizione degli altri veicoli. I diversi nodi della rete faranno riferimento in modo specifico a questi server per conoscere le informazioni ricercate.

L'elezione dei location server può essere effettuata seguendo due possibili strategie:

- **Hash-based LS**
  Questa tecnica fa uso di una particolare funzione di hash *H(x)*, come SHA-1 o MD5, nota a tutti i nodi della rete, per mappare l'identificativo di un dato nodo con altri nodi o regioni della rete che fungono da location server per quello specifico nodo. Allo stesso tempo questa funzione di hash divide la rete in regioni gerarchiche. In ogni regione, un veicolo, sfruttando la stessa funzione di hash, viene scelto come location server.

- **Quorum-based LS**
  In questa strategia la posizione di ogni nodo viene inviata periodicamente ad un set di veicoli che formano il quorum e che saranno usati come location

server. E' un esempio di questa tecnica il protocollo RLSM (*Reactive Location Service*) [17].

Per concludere questo aspetto delle VANET, citiamo alcuni dei più interessanti protocolli per l'implementazione del Location Service: ILS (*Intersection Location Service*) [18], VLS (*Vehicle Location Service*) [19] e, per finire, MBLS (*Map-Based Location Service*) [20].

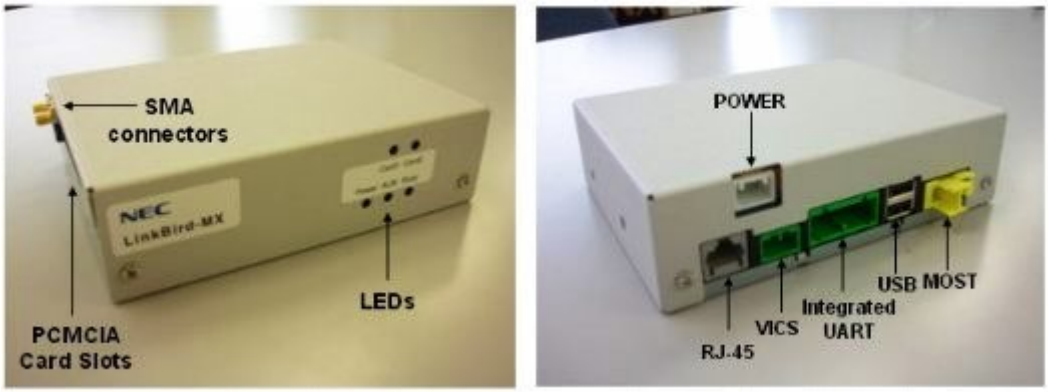


Figura 2.8: LinkBird-MX sviluppato da NEC e C2CCC (fonte: [21])

2.8 Dispositivi sviluppati

L'unico prototipo funzionante ad oggi sviluppato è il *LinkBird-MX* (Figura 2.8) realizzato da C2CCC su piattaforma Linux in collaborazione con NEC. Dal punto di vista hardware, il LinkBird-MX monta un processore MIPS a 64bit, 512MB di memoria NANDFlash e 128MB di memoria SDRAM. Esso implementa lo standard IEEE 802.11p e contiene, oltre alle porte Ethernet e USB, anche un GPS, due antenne e due slot PCMCIA per le schede WLAN. Per quanto riguarda il comparto software, questo dispositivo fa affidamento al kernel Linux 2.6.19 ed esegue lo stack NEC C2X-SDK Communication System (Figura 2.9) e API. Si tratta dell'implementazione software dello stack protocollare per le OBU e le RSU. L'implementazione, descritta e minuziosamente raffigurata in ogni suo componente in [22], comprende sia le funzioni di rete di base e sia molte caratteristiche avanzate come il meccanismo per il miglioramento dell'efficienza e dell'affidabilità della trasmissione dei dati, l'invio periodico dei beacon con controllo di potenza distribuito, il Location Service basato su protocollo RLS (Reactive Location Service), la protezione crittografica e l'autenticazione.

Sono presenti tre interfacce indicate nella Figura 2.9 da tre blocchetti con le diciture Data, Management e IP. Le prime due sono di fondamentale importanza in quanto permettono di configurare un Location Service, gli pseudonimi, le credenziali di sicurezza, inviare/ricevere messaggi e leggere/scrivere la posizione geografica del veicolo. La terza, l'interfaccia IP, consente l'esecuzione di applicazioni basate sul protocollo di rete IPv4/v6.

I due blocchetti con le sigle IC e IH indicano rispettivamente l'*Information Connector* (consente lo scambio asincrono di messaggi tra i vari strati) e l'*Information Handler* (aggrega le informazioni e le imposta all'interno dei beacon).

L'instradamento dei pacchetti viene effettuato mediante l'utilizzo di un protocollo *Geocast* basato sulla posizione dei veicoli e su contesa. E' inoltre possibile usare dei file di configurazione per il settaggio delle diverse funzionalità messe a disposizione dalle varie componenti dello stack protocollare (si può ad esempio scegliere se attivare o meno il tracciamento dei pacchetti o la registrazione degli eventi su file di log). Grazie a questa particolare gestione ad hoc dei parametri mediante file l'utente può, ad esempio, effettuare una rapida variazione della configurazione a seconda delle diverse fasi di studio e/o sviluppo delle applicazioni.

Ad oggi, il LinkBird-MX unitamente al C2X-SDK risulta essere senza dubbio la più importante messa in pratica delle caratteristiche e delle proprietà funzionali, fino a quel momento ideologiche, delle Vehicular Ad-Hoc Network.

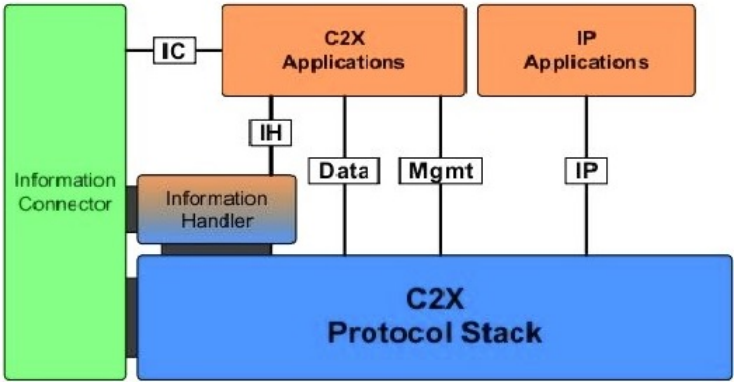


Figura 2.9: Architettura Car-2-X SDK

2.9 Strumenti utili per le simulazioni

Per poter investigare su ogni possibile aspetto e problematica inerente l'applicazione pratica dei principi delle VANET, è indispensabile passare prima da una fase di studio intermedia: la simulazione virtuale. Questa permette agli sviluppatori di valutare la qualità delle loro ricerche senza incorrere in gravosi costi legati eventualmente ad una reale sperimentazione pratica, che oltretutto risulterebbe lenta e non facilmente ripetibile come invece può esserlo una simulazione virtuale. E' quindi veramente importante cercare di costruire un ambiente virtuale più realistico possibile. Solo così sarà possibile avere un'idea di come potrebbe effettivamente funzionare in uno scenario reale quel dato applicativo o protocollo sviluppato. Esiste un'ampia gamma di simulatori software in commercio che permettono sia la simulazione della comunicazione tra veicoli mediante scambio di beacon, OMNeT++ [23] e NS-2 [24] ne sono un esempio, e sia la creazione di un ambiente virtuale nel quale i veicoli possono effettivamente spostarsi su dei tragitti cittadini reali seguendo delle regole predefinite durante il loro spostamento. Questi ultimi sono i cosiddetti simulatori

di traffico e nel nostro caso di studio abbiamo fatto uso di uno tra i più importanti e famosi: SUMO (*Simulation of Urban MObility*) [25].

Per poter mettere in atto un corretto caso di studio è indispensabile far interagire i due tipi di simulatori.

2.9.1 SUMO (Simulation of Urban MObility)

SUMO è un simulatore di traffico sviluppato dal reparto ITS del *German Aerospace Center*, rilasciato con licenza Open nel 2001 e utilizzato ampiamente in molti progetti di ricerca internazionali. Ad esempio, SUMO è stato usato per fornire la previsione del traffico per le autorità della città di Colonia (Germania) durante la visita del Papa nel 2005 e durante il Campionato Mondiale di Calcio del 2006. Questo importantissimo tool è attualmente utilizzato in numerosi progetti di ricerca, come VABENE [26] e VEU [27], ed è oggetto di costanti aggiornamenti e migliorie grazie all'imponente apporto della comunità internazionale di utenti che, unitamente ai maggiori rappresentanti del mondo dell'industria e della comunità scientifica, annualmente si riunisce in una conferenza per discutere, tra le altre cose, dello stato di avanzamento della ricerca e dello sviluppo di nuove funzionalità in SUMO.

Questo tool permette l'implementazione di uno scenario di traffico personalizzato. Per procedere con l'utilizzo di SUMO è sufficiente scaricare ed eseguire dal sito ufficiale il pacchetto precompilato compatibile con il proprio ambiente di sviluppo (Windows, Linux o Mac OS) [28]. All'interno del package di SUMO sono presenti una serie di plugin che permettono, tra le altre cose, la generazione di rotte stradali, la simulazione mediante interfaccia grafica oppure l'importazione di una rete stradale creata dall'utente mediante programmi esterni appositi, come Open Street Map [29].

La descrizione dell'ambiente virtuale avviene mediante il passaggio di una serie di file opportunamente definiti dall'utente al simulatore. Questi file comprendono: informazioni sulla rete stradale (file con estensione .nod.xml e .edg.xml), informazioni sul traffico e sulle rotte dei veicoli (file con estensione .rou.xml) e, se sono richieste, informazioni addizionali sulle regole che definiscono il movimento consentito del traffico veicolare sulla rete stradale e sulle connessioni tra le corsie stradali (file con estensione .con.xml e .typ.xml).

SUMO implementa anche il cosiddetto IDM (*Intelligent Driver Model*), che risulta essere il modello di mobilità più usato in questo ambito di ricerca, secondo il quale è possibile modellare in modo accurato e realistico il comportamento dei conducenti delle automobili nello scenario stradale simulato.

Uno degli output generati da SUMO è il file XML relativo alle *tracce* dei veicoli in movimento durante la simulazione. La dimensione di questo file è variabile ed è dipendente dalla "complessità" della simulazione stessa. Dal punto di vista pratico, questo file descrive alcune grandezze che identificano istante per istante le proprietà del veicolo. Ogni traccia include l'informazione sulla posizione del veicolo, sulla velocità del veicolo e sull'identificativo del veicolo. La posizione, in modo particolare, può essere rappresentata sia mediante coordinate spaziali dipendenti dalla proiezione

geografica e sia per mezzo di valori numerici che rappresentano la latitudine e la longitudine del veicolo nella situazione in esame.

Le informazioni ottenute dalla simulazione possono essere tradotte in diverse forme grazie all'integrazione di SUMO con numerosi plugin ed estensioni. Un plugin particolarmente importante è *TraceExporter*. Questo permette di tradurre l'output di SUMO in differenti formati appetibili da molti altri tool. Ad esempio, se fossimo interessati all'importazione delle tracce dei veicoli su una base di dati per la loro successiva elaborazione, basterebbe utilizzare TraceExporter e scegliere come output un file di tipo CSV (Comma Separated Values). Invece, per quanto concerne le estensioni, possiamo citare Veins (tool utile per rendere possibile la comunicazione tra SUMO e OMNeT++) o TraCI (tool che consente di ottenere dei valori relativi agli oggetti simulati e di interagire con essi per manipolarne il comportamento on-line).

In conclusione, è possibile affermare che, a partire dalla data della sua pubblicazione nel lontano 2001 e grazie alla comunità di sviluppatori che lo hanno supportato e migliorato costantemente, SUMO è sempre stato e continua ad essere un software di punta usato dalla maggior parte dei gruppi di ricerca che si sono affacciati nel mondo delle VANET.

Capitolo 3

Beaconing

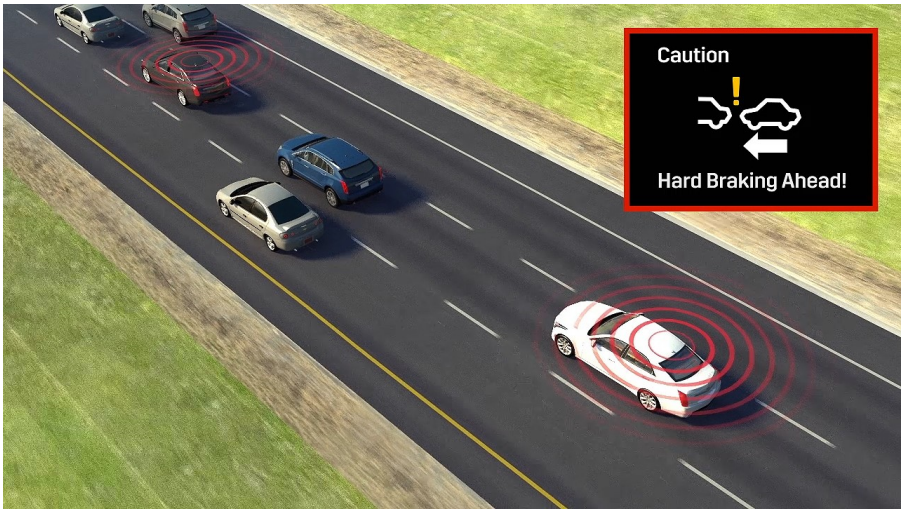


Figura 3.1: Esempio di trasmissione messaggio di sicurezza

Nei capitoli precedenti si è fatto spesso riferimento ad un processo di particolare importanza per una rete come la VANET che basa la sua filosofia sul concetto di salvaguardia della sicurezza degli utenti sulle strade di tutto il mondo. Si tratta del processo di *beaconing*.

Tra le applicazioni ed i servizi che possono essere sviluppati ed attivati sulle VANET, il beaconing è il più importante poiché da esso dipende il corretto svolgimento delle principali funzionalità di queste reti.

I messaggi generati da questo servizio sono chiamati beacon. I beacon sono trasmessi con una certa frequenza da tutti i nodi della rete ai propri vicini e contengono una serie di informazioni che permettono la comunicazione di dati utili per la gestione delle operazioni connesse alla sicurezza stradale.

Purtroppo il beaconing non è esente da problemi connessi sia alla sua implementazione, ad esempio è indispensabile decidere la frequenza alla quale generare i beacon o la loro dimensione, sia alla protezione delle informazioni trasmesse. Quello della protezione dei dati è un ostacolo di non poco conto. E' indispensabile definire una strategia efficiente che miri ad ottenere la protezione dei dati sensibili contenuti

all'interno di questi beacon per evitare che utenti maliziosi possano usarli per scopi non proprio leciti.

In questo capitolo viene affrontato il processo di beaconing, valutate le necessità e le problematiche ad esso connesse, considerato il processo di geolocalizzazione e, per finire, discusse in modo più ampio le soluzioni presenti in letteratura per il miglioramento di questo processo.

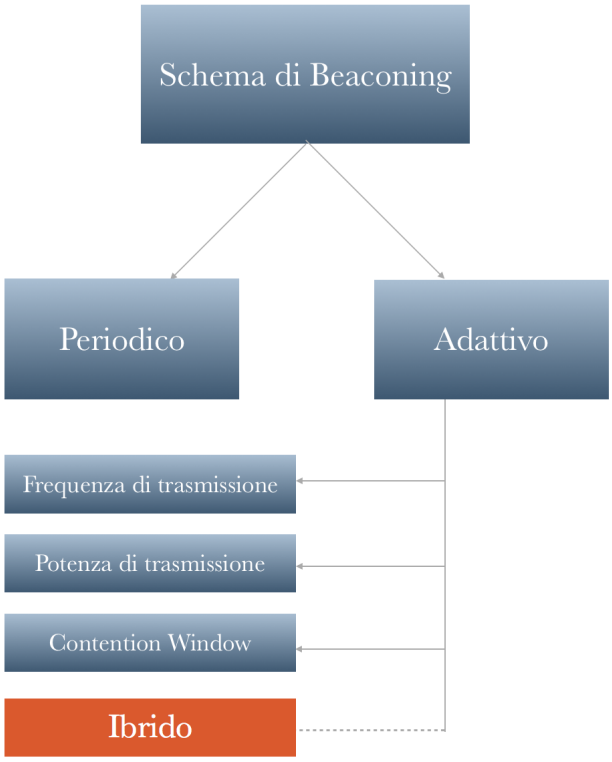


Figura 3.2: Classificazione delle tecniche di beaconing

3.1 Proprietà

Il Beaconing è un semplice meccanismo utilizzato nelle reti veicolari per la trasmissione di piccoli messaggi, chiamati beacon o CAM (*Cooperative Awareness Messages*), aventi una dimensione di qualche centinaio di byte che ricoprono un ruolo speciale ovvero quello di fornire informazioni sul veicolo che li ha generati ai veicoli limitrofi entro un raggio di circa 300 metri.

La generazione di questi messaggi è totalmente indipendente da ogni possibile applicazione, la trasmissione avviene senza alcuna coordinazione tra i nodi e con una frequenza molto elevata, ad esempio 10 messaggi al secondo. La frequenza con la quale questi pacchetti vengono generati e immessi sulla rete non è fissata e numerosi studi sono stati condotti per cercare di trovare una grandezza fissa ottimale che si adatti ai diversi scenari o un algoritmo adattivo che vari il numero di beacon generati a seconda dei vari contesti applicativi e dei diversi parametri (Figura 3.2). Queste

ultime due soluzioni danno vita a due differenti schemi di gestione del processo di beaconing chiamati *beaconing periodico* e *beaconing adattivo*.

I messaggi generati dal processo di beaconing sono talvolta suddivisi in 3 categorie a seconda del livello dello stack di rete al quale fanno riferimento le "applicazioni" che li hanno generati. Il beaconing di Tipo 1 viene usato dal livello MAC per lo scambio dei parametri di rete, il beaconing di Tipo 2 è utilizzato dal livello di rete e, per concludere, quello di Tipo 3 dal livello applicazione.

I beacon sono trasmessi sul canale DSRC e sono gestiti a livello MAC come dei pacchetti broadcast. A differenza di altri tipi di messaggi, questi non vengono mai ritrasmessi dai nodi che li ricevono ed essendo vitali per le applicazioni di sicurezza devono registrare minimi ritardi di trasmissione e soprattutto devono raggiungere con successo la loro destinazione.

Risulta particolarmente importante la stima di un valore che permetta di sapere per quanto tempo un beacon debba "restare in vita" sul canale affinché tutti i veicoli interessati possano sentirlo e la formula che segue (Equazione 3.1) permette di calcolare questa sorta di TTL (Time To Live) in relazione ad alcuni parametri di rete:

$$T_{\mathrm{s}} = T_{\mathrm{h}} + \frac{S}{R} \tag{3.1}$$

dove T_{h} è il tempo di trasmissione dell'header del pacchetto, S è la dimensione in bit di un beacon e R è la frequenza di trasmissione dei dati in $^{bit}/_{s}$ [30]. Come si può notare dalla formula, il tempo T_{s} dipende fortemente dal rapporto $^{S}/_{R}$ ed in particolare dal data rate R. Tuttavia se da un lato l'aumento della frequenza di trasmissione dei dati R apporta un miglioramento al T_{s}, dall'altro può avere anche effetti negativi dovuti alle interferenze o ai possibili errori di ricezione dei pacchetti. Purtroppo lo standard WAVE non mette a disposizione strategie particolari come la ritrasmissione dei pacchetti atte ad assicurare il packet delivery, anche se in letteratura sono presenti delle proposte molto interessanti come [31] che presenta una strategia composta da tre fasi (assegnazione di priorità ai pacchetti - gestione della congestione di rete - gestione adattiva della frequenza di trasmissione dei dati) per una gestione efficiente e ad hoc dei dati prodotti dalle varie applicazioni delle Vehicular Ad Hoc Network.

Il processo di beaconing causa perciò un grosso carico sulla rete e la banda rubata da questo può comportare un aumento delle collisioni dei frame MAC ed alla congestione della rete con conseguenti rallentamenti nei trasferimenti dei dati.

La dimensione dei beacon è variabile e dipende dalla quantità di informazioni inserite al loro interno. Questo può portare a dei "comportamenti" inaspettati della rete e causare dei grossi problemi di scalabilità.

Il traffico generato da questo processo non può essere trattato come un normale traffico appartenente ad un particolare servizio sviluppato sulla rete VANET perché è esso stesso il principale servizio delle reti VANET. Questo è il principale motivo per cui a queste informazioni viene attribuita una priorità maggiore rispetto alle altre.

3.2 Geolocalizzazione

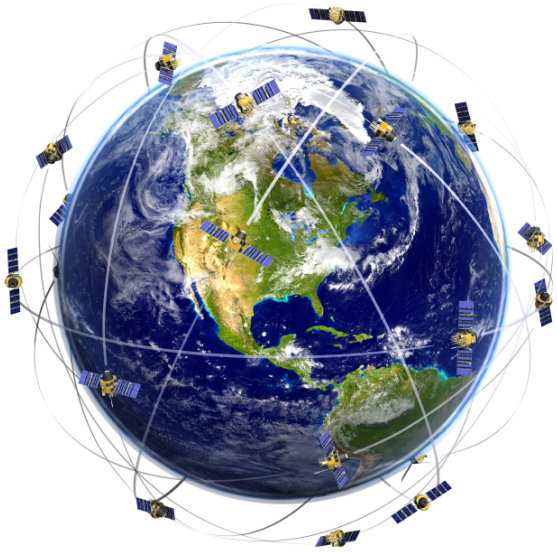


Figura 3.3: Rappresentazione grafica del sistema GPS

Una delle informazioni contenute nei messaggi di sicurezza scambiati tra i veicoli è la posizione geografica. Affinché questi dati possano essere prelevati, elaborati e poi inseriti negli appositi campi dei pacchetti, è indispensabile che il sistema preveda un meccanismo veloce ed efficiente per la localizzazione del veicolo sul globo terrestre.

Un sistema satellitare globale di localizzazione GNSS è un sistema per la radionavigazione che integra le funzioni dei satelliti di posizionamento del sistema GPS (*Global Posistioning System*) con altri satelliti e stazioni terrestri per consentire la tracciabilità di un oggetto durante il suo spostamento con una elevata precisione.

I più importanti sistemi GNSS sono *GPS*, *GLONASS (GLObal Orbiting NAvigation Satellite System)* e *GALILEO*. Tra tutti il GPS è senza dubbio il più utilizzato in quanto consente il rilevamento, oltre che della posizione, anche della velocità del ricevitore relativa alla Terra, attraverso la determinazione dell'effetto doppler sui radiosegnali. Purtroppo le informazioni generate dal GPS non sono affidabili ed accurate al 100% e nelle reti VANET, al fine di ottenere dati sulla posizione dei nodi con il minimo errore, si fa talvolta affidamento al cosiddetto sistema DGPS (*Differential GPS*). Il grado di errore con l'ausilio del sistema DGPS è compreso tra 1 e 5 metri, usando solo il GPS si potrebbe invece registrare un errore minimo di 5 metri nel caso migliore e fino addirittura a 40 metri nel caso peggiore. L'idea che sta alla base del DGPS è quella di rimediare agli errori del GPS e a eventuali disturbi del segnale captando le trasmissioni di almeno quattro satelliti e combinando le informazioni ricevute con quelle contenute in un archivio elettronico una volta nota la reale distanza Terra-satellite.

Per quanto riguarda il funzionamento di un reale sistema GNSS, ogni ricevitore è in grado di ricevere dei messaggi NAV (*Navigation Message*) dai satelliti in orbita ai quali viene assegnato un codice di distribuzione univoco pubblicamente noto. Per questo motivo questo sistema è particolarmente vulnerabile: un attaccante potrebbe

fingersi un satellite autorizzato e fornire segnali manipolati identici a quelli trasmessi dal satellite reale [32]. Questa manipolazione può essere compiuta in due fasi:

- disturbo dei reali segnali GNSS (*Jamming*);
- trasmissione dei messaggi manipolati (*Spoofing*).

Per limitare le minacce relative alla sicurezza di questo sistema di rilevazione terrestre bisognerebbe garantire l'autenticità dei messaggi e della loro origine attraverso uno *schema di autenticazione* che dia garanzie sull'entità che ha generato il segnale e di uno schema di *protezione dell'integrità dei dati* che eviti possibili manipolazioni dei contenuti dei messaggi attraverso attacchi di spoofing.

L'autenticazione dei messaggi trasmessi dai satelliti mediante uno schema di crittografia a chiave pubblica è senza dubbio un prerequisito di particolare importanza ed il sistema di rilevazione europeo GALILEO, prima menzionato ed entrato in servizio nel dicembre del 2016, ne prevede l'uso attraverso il servizio NMA (*Navigation Message Authentication*).

3.3 Limitazioni e necessità

Il beacon, come detto più volte, è un pacchetto generato in modo automatico da ogni nodo trasmesso in broadcast sul Control Channel DSRC strettamente connesso alle applicazioni di sicurezza. La quantità di pacchetti immessi sul canale in una specifica area della rete dipende fortemente dalla densità e dalla distribuzione dei veicoli, per definizione non uniforme, e la conseguente congestione di questo canale può portare a delle gravissime problematiche che si riflettono poi sui ritardi registrati da questi pacchetti.

Una rete dalle dimensioni importanti come la rete VANET deve prevedere un sistema di controllo della congestione che valuti istante per istante il carico mediante la definizione di opportuni parametri e che, in caso di forte sovraccarico, agisca provvedendo alla diminuzione del numero di dati immessi sulla stessa rete dai nodi che ne fanno parte attraverso la variazione della frequenza di generazione/trasmissione dei beacon. In questo modo è possibile contrastare un eventuale calo delle prestazioni della rete che potrebbe causare una diminuzione della probabilità di ricezione dei beacon da parte dei vari nodi [33].

Lo standard IEEE 802.11p (Capitolo 2, Paragrafo 2.3) ha avanzato una possibile strategia per la gestione della saturazione del canale: quando il tasso di occupazione del canale di trasmissione supera la soglia del 50% tutti i messaggi vengono bloccati eccezion fatta per quelli aventi priorità massima. Purtroppo questa non risolve il problema della congestione del mezzo di comunicazione causata dalla trasmissione periodica dei beacon. A tal proposito un gran numero di simulazioni ed esperimenti sono stati condotti poiché risolvendo questo grattacapo verrebbe difatti annullato un grosso limite di una rete VANET.

Nelle applicazioni di sicurezza un alto tasso di generazione dei beacon può senza dubbio rendere migliore l'accuratezza delle informazioni. Il carico del canale causato dal processo di beaconing potrebbe essere gestito mediante la variazione della potenza di trasmissione che causerebbe una variazione del range di trasmissione del singolo nodo. In [34] viene ad esempio proposta una soluzione che fa uso del TDMA (Time Division Multiple Access) per controllare la potenza di trasmissione del veicolo obbligando il numero di veicoli circostanti a rimanere entro un intervallo prestabilito e mantenendo il carico del canale entro un certo intervallo di valori.

La strategia di utilizzo dei beacon per scoprire e mantenere le relazioni tra nodi limitrofi ha guadagnato nel corso degli ultimi anni sempre più credito sia per le comunicazioni unicast che broadcast. Tuttavia la strategia qua considerata fonda la sua natura costruttiva sulla presenza di una tabella, chiamata *Neighbor Table*, le cui entry, che descrivono lo stato del vicinato di un nodo, vengono rinfrescate in continuazione associandovi un "age threshold" che una volta trascorso causa l'invecchiamento delle informazioni e l'eliminazione dalla tabella. Questa "soglia di invecchiamento" dovrà avere un valore molto basso per far si che le entry all'interno della tabella, aggiornandosi frequentemente con le informazioni prelevate dai beacon ricevuti, riflettano fedelmente la struttura topologica della rete nell'intorno di ogni nodo.

3.4 Lavori presenti in letteratura

Quello delle VANET è un importante campo di ricerca in continua evoluzione e argomenti come il routing e il beaconing sono costantemente sotto la lente di ingrandimento dei numerosi gruppi di ricerca che nel corso degli anni hanno proposto un'ampia gamma di soluzioni per la risoluzione delle diverse problematiche inerenti all'implementazione di queste reti.

Vengono di seguito presentate alcune delle soluzioni avanzate dalla comunità scientifica per l'efficientamento del processo di beaconing e dei servizi ad esso affini.

Numerosi approcci sono stati presentati per la riduzione del numero di beacon sulla rete [35]. Molti di questi adattano il beacon rate in relazione alla tipologia dell'ambiente nel quale si trova un certo nodo sulla base di alcune proprietà come densità o velocità dei veicoli. Altri ancora propongono degli schemi di desincronizzazione dei nodi per prevenire la collisione dei messaggi sul canale di comunicazione ed evitare quindi successive ritrasmissioni che accrescono l'overhead della rete [36].

In [37] vengono valutati diversi schemi di trasmissione multi-hop broadcast dei pacchetti per la stima di un data rate ottimale per assicurare una corretta consegna fino ai destinatari. Vengono in particolare valutati mediante simulazioni tre diversi protocolli per la propagazione delle informazioni sulla rete che prendono il nome di DAD (*Distribution-Adjusted Distance Protocol*), SBA (*Scalable Broadcast Algorithm*) e MPR (*Multi-Point Relaying*) poiché anche da questi dipende il contenuto dei beacon generati e trasmessi dai nodi. Ad esempio, facendo uso dei due protocolli SBA e MPR i nodi dovranno inserire all'interno del beacon oltre al loro ID anche la lista aggiornata dei nodi facenti parte del vicinato.

In [38] viene proposto uno schema per la generazione di beacon a dimensione fissa attraverso l'uso di Bloom Filter per la riduzione dell'overhead che questo processo apporta all'intera rete e vengono condotte delle simulazioni per provarne l'efficacia considerando tre fattori: affidabilità, overhead per le ritrasmissioni e overhead apportato dai beacon sulla rete. I Bloom Filter sono ampiamente utilizzati in diversi campi dell'ingegneria ed in questo contesto permettono di ridurre la complessità dell'algoritmo di consegna dei beacon.

Infine in [39] viene condotta una simulazione realistica per calcolare la probabilità di consegna dei beacon, definita tramite la sigla PBD, con lo standard *IEEE 802.11p* in scenari differenti attraverso l'uso di simulatori come SUMO, Veins e OMnET++, al fine di confrontare i risultati ottenuti con il modello matematico ideale e verificarne la validità.

Capitolo 4

Sicurezza

La *sicurezza* è un concetto di fondamentale importanza per qualsiasi sistema di comunicazione. Non si può pensare di sviluppare le applicazioni di rete o i protocolli senza considerare prima questo aspetto. Affinché le VANET diventino una realtà è indispensabile che una appropriata architettura di sicurezza venga definita per la protezione da eventuali attacchi.

Informazioni di vitale importanza come quelle connesse alla sicurezza stradale, non dovrebbero poter essere né modificate e né cancellate da un potenziale attaccante. Allo stesso tempo, la privacy è un problema che deve essere preso in considerazione. I dati relativi ai veicoli e ai loro conducenti devono essere scambiati in modo sicuro per evitare che utenti maliziosi possano venirne in possesso per condurre eventuali attacchi o semplicemente per conoscerne il contenuto e spiare gli utenti inconsapevoli durante i loro spostamenti.

Tutte le soluzioni ammissibili per la gestione di queste problematiche devono considerare un ulteriore ostacolo: il costo computazionale. Ogni nodo riceve una grossa mole di informazioni che devono essere manipolate in modo sicuro e per questo motivo le strategie da adottare devono scontrarsi con le limitate capacità di elaborazione dei dispositivi a bordo dei veicoli che per ragioni di costo non possono avere caratteristiche di alto livello.

Purtroppo lo sviluppo di un sistema sicuro, nel caso delle VANET, è un ostacolo di non poco conto. La dinamicità di queste reti, la trasmissione delle informazioni in ambiente aperto e l'utilizzo di un canale di comunicazione wireless le rendono particolarmente vulnerabili. Inoltre, a causa delle diverse leggi sulla sicurezza e la privacy dei vari paesi nel mondo, non sarà così semplice sviluppare uno standard univoco che possa mettere tutti d'accordo.

In questo capitolo verranno affrontati i problemi connessi alla sicurezza delle reti VANET partendo dai requisiti per la sicurezza di queste reti e continuando con le possibili tipologie di attacco e di attaccanti e il problema della tutela della privacy.

4.1 Requisiti di sicurezza

In accordo con lo standard *X.800* definito dall'ITU-T, una minaccia per un sistema di sicurezza può riguardare la distruzione o la rimozione di informazioni, la corruzione o la modifica di dati, il prelievo di dati o l'interruzione di un servizio di rete. La *RFC2828* pubblicata nel 2000 dall'IETF fornisce un'ulteriore definizione di minaccia e la denota come una potenziale azione, circostanza o evento di violazione della sicurezza di un sistema che potrebbe causarne il danneggiamento. In altre parole, una minaccia è una potenziale insidia che potrebbe sfruttare delle vulnerabilità del sistema.

I requisiti per la sicurezza nelle VANET corrispondono in parte ai servizi di sicurezza definiti nello standard X.800:

- **Autenticazione**
  Assicura che le informazioni contenute in un messaggio siano state generate da entità affidabili. Nelle VANET è molto importante questa proprietà in quanto i nodi fanno un forte affidamento ai dati contenuti nei beacon ricevuti. I servizi di autenticazione sono divisi in due categorie: *autenticazione dell'origine dei dati e autenticazione dell'identità di un nodo*.

- **Controllo degli accessi**
  Controlla chi può avere accesso ad una risorsa, sotto quali condizioni può essere acceduta quella risorsa e come può essere acceduta da diverse classi di utenti. In questo modo si provvede alla protezione dagli accessi non autorizzati alle risorse computazionali, ai dischi o ai collegamenti di rete.

- **Riservatezza dei dati**
  Questo servizio protegge i dati da accessi non autorizzati durante il loro spostamento dalla sorgente alla destinazione. Le informazioni trasportate dai beacon devono essere accedute solo da coloro i quali ne hanno il diritto per evitare eventuali manomissioni. La crittografia viene spesso utilizzata per offrire la riservatezza delle informazioni.

- **Integrità dei dati**
  Il contenuto dei messaggi deve essere protetto per evitare che un'entità non fidata possa alterarlo inavvertitamente e questo servizio si assicura che ciò avvenga.

- **Non ripudio**
  Il nodo che ha generato ed inviato un beacon non può negare di averlo fatto. In questo modo si ottiene la sicurezza sull'autore dei dati trasmessi il quale non può negare il suo coinvolgimento nella comunicazione. Inoltre quando un nodo A riceve un messaggio "fasullo" dal nodo B, A può usare il messaggio ricevuto per accusarlo della sua non veridicità e convincere gli altri nodi della rete che B è un utente malizioso.

- **Disponibilità**
  Il sistema deve rimanere operativo anche in caso di guasti o attacchi e prevedere

dei meccanismi di ripresa dell'attività in seguito a problemi di qualsiasi natura che possono verificarsi durante il suo normale funzionamento.

- **Tracciabilità**
  Deve essere sempre possibile identificare tutti i nodi e i messaggi scambiati nella rete. Il problema della tracciabilità è legato al problema dell'identificazione delle responsabilità. Se un nodo dovesse causare un incidente deve esser possibile identificarlo per sottoporlo alle conseguenze.

Oltre ai servizi appena indicati bisogna considerare un altro aspetto di particolare rilevanza per le reti VANET: la *privacy*. Le informazioni personali trasmesse dai nodi mobili attraverso i beacon devono essere opportunamente protette per evitare che utenti non autorizzati possano accedervi inavvertitamente e prelevarle. Un problema che sorge in questo contesto è quello della tracciabilità degli spostamenti dei nodi della rete e per contrastarlo diversi schemi di protezione dei messaggi sono stati proposti [40, 41]. Purtroppo molti di questi schemi presentano delle prestazioni inadeguate per il contesto applicativo per il quale sono stati sviluppati.

Non tutti i messaggi scambiati tra i veicoli necessitano di essere protetti tramite cifratura in quanto non tutti i messaggi contengono dati sensibili e per tale ragione è sufficiente talvolta adottare un semplice meccanismo di autenticazione simmetrico o asimmetrico. Gli schemi di autenticazione simmetrica sono di raro utilizzo nel contesto delle VANET a causa della scarsa flessibilità di impiego anche se, rispetto a quelli asimmetrici, richiedono delle potenze di calcolo ridotte ed offrono maggiore resistenza agli attacchi di cripto analisi.

La validazione dei dati scambiati tra i nodi può essere effettuata utilizzando uno schema che unisca un sistema di autenticazione con un sistema di sensori a bordo dei veicoli [42]. In questo modo i veicoli possono elaborare i dati captati dai sensori per migliorare la qualità delle informazioni trasmesse e per accertarsi della veridicità dei dati ricevuti dagli altri nodi. L'impiego massiccio di sensori e di tecniche di studio che sfruttino la ridondanza dei dati per la verifica della loro veridicità possono senza dubbio limitare le vulnerabilità delle reti VANET.

La protezione della privacy degli utenti e la sicurezza della rete sono due proprietà fondamentali che potranno senza dubbio incoraggiare le persone ad entrare far parte del mondo VANET e le aziende ad investire per la futura crescita e lo sviluppo di queste reti e delle relative applicazioni.

4.2 Componenti fidate e certificati

L'architettura di sicurezza deve far affidamento ad una terza parte fidata o autorità che gestisca le identità, le credenziali di accesso alla rete e le chiavi per la cifratura di tutti i nodi in una certa regione della rete. Da adesso in poi faremo riferimento a questa entità come la *Certification Authority (CA)*.

Ogni veicolo facente parte della rete è rappresentato mediante un identificativo univoco U ed una coppia di chiavi, pubblica k_u e privata K_u, per le operazioni di cifratura. $Cert_y\{K_u,\ A_u\}$ è un certificato rilasciato da una CA Y che associa ad un

certo utente una lista di attributi A_u. In assenza di questo certificato i nodi non possono prendere parte alle operazioni della rete.

I certificati non possono avere una validità infinita e quindi le CA devono preoccuparsi di gestirne il ciclo di vita ed eventualmente effettuarne la revoca. La revoca del certificato è un'operazione che invalida un certificato, precedentemente rilasciato da una certa CA, prima della sua naturale scadenza. Le ragioni per cui si può voler procedere con la revoca sono tante. Ad esempio si potrebbe voler revocare un certificato per ragioni amministrative come il cambio della regione di appartenenza di un veicolo oppure a seguito di un furto dei dati personali utili per le operazioni di cifratura come la chiave privata.

L'accesso istantaneo alle informazioni di revoca è indispensabile per un funzionamento sicuro della rete: ogni nodo deve poter accedere in qualsiasi momento a questi dati per sapere se può fidarsi o meno della controparte che gli ha inviato un messaggio. Purtroppo, a causa della natura costruttiva e dell'ampiezza delle VANET, lo sviluppo e il successivo inserimento di un sistema che gestisca in modo efficiente la distribuzione delle informazioni di revoca dei certificati non è un'operazione semplice. E' chiaro che una buona parte del lavoro dovrebbe essere svolto dalle RSU, ma che cosa accadrebbe se questa infrastruttura di rete ad un certo punto dovesse smettere di funzionare? Questo è senza dubbio un problema di non poco conto. Nel contesto delle Vehicular Ad Hoc Network [43] è un importante punto di partenza per il problema della revoca dei certificati, anche se si limita soltanto a considerare la distribuzione delle CRL (*Certificate Revocation List*) e non le modalità pratiche per realizzarla mentre [44] rappresenta uno dei primi studi effettuati relativamente a questa problematica e propone 3 possibili soluzioni per coprire altrettanti possibili scenari. Infine [45] e [46] rappresentano rispettivamente gli standard per gli attributi dei certificati e per le CRL.

Quanto discusso finora però non considera le tecniche da adottare per aiutare le CA ad individuare gli utenti non autorizzati o maliziosi per procedere con la revoca dei certificati. [47] propone due diverse strategie che prendono il nome di LEAVE (*Local Eviction of Attackers by Voting Evaluators*) e Stinger. La prima soluzione si basa su un sistema di comunicazione broadcast tra i nodi e su una *accusation list* che permette ad un nodo che identifica un utente malizioso di inoltrare un messaggio per informare i nodi limitrofi i quali a loro volta provvedono ad inserire l'identità di quel nodo nella lista e ad inoltrare il messaggio. Nel momento in cui il nodo sospetto riceve un certo numero di voti "sfavorevoli" la sua identità viene aggiunta ad una blacklist locale che sarà poi inoltrata alla CA di quella regione la quale provvederà alla verifica e all'eventuale revoca del certificato. La seconda proposta chiamata Stinger basa il suo funzionamento sul concetto di suicidio già proposto in precedenza da altri. In questa soluzione un nodo può accusare un altro nodo di essere un utente illegittimo e causarne l'uscita dalla rete ma potrà farlo solo a patto di pagare l'accusa fatta con la propria eliminazione, da qui il concetto di *suicidio*.

I nodi sono equipaggiati con delle componenti fidate che svolgono una duplice funzionalità: cifratura delle informazioni e memorizzazione dei dati degli utenti. Il possesso delle credenziali di accesso alla rete non garantisce che un certo nodo sia affidabile e sicuro. Virtualmente ogni nodo nella rete può far venire meno la funzionalità di un protocollo attraverso un attacco e limitare o bloccare la comunicazione

tra i nodi. La distinzione tra un problema proprio della rete e un attacco non è un'operazione semplice. La presenza di servizi di sicurezza unitamente ad un robusto sistema di rilevamento degli errori di comunicazione sono assolutamente necessari per rendere sicura la trasmissione dei dati nella rete. Nel caso di un problema di perdita dei pacchetti delle opportune contromisure devono essere intraprese come la ritrasmissione da parte dei nodi sorgente o il cambio delle rotte per evitare che i dati seguano percorsi "compromessi".

4.3 Attacchi alle VANET

Nel contesto delle Vehicular Ad Hoc Network vi è un numero non indifferente di minacce e quindi di attacchi ad esse connessi che non possono essere ignorati, ma prima di procedere alla loro classificazione è bene individuare le diverse categorie di attaccanti [48].

4.3.1 Classificazione degli attaccanti

Gli attaccanti possono essere suddivisi in diverse categorie in base ad una serie di proprietà che li caratterizzano come la "posizione" nella rete o l'intenzionalità dell'attacco. Vengono di seguito proposte tre categorie di attaccanti:

- ***attivi e passivi***
  Gli attaccanti attivi si mimetizzano con gli altri nodi della rete e si preoccupano di iniettare informazioni fasulle o modificare il contenuto dei messaggi in transito sulla rete mentre i secondi, quelli passivi, si limitano ad intercettare il traffico per apprendere le informazioni contenute nei beacon che saranno utili successivamente per condurre degli attacchi. Gli utenti maliziosi possono agire sia singolarmente che a gruppi per compiere degli attacchi più sofisticati e disastrosi. Ad esempio, una serie di nodi coordinati potrebbero iniettare nella rete delle informazioni fasulle relative ad un incidente avvenuto da qualche parte per smistare il traffico veicolare verso altre zone e liberare quella zona specifica per compiere un furto o qualche altro atto illegale o terroristico (*Bogus Information*).

  Gli attaccanti passivi, a differenza di quelli attivi, sono molto difficili da individuare in quanto il traffico sulla rete fluisce in modo apparentemente normale e né il mittente e né il destinatario possono facilmente accorgersi che un terzo vi ha acceduto. In genere questo tipo di attacchi vengono combattuti non sviluppando algoritmi per individuarli ma utilizzando degli schemi ad hoc, come la cifratura, che permettono di offuscare il contenuto dei messaggi.

- ***interni ed esterni***
  Qualsiasi nodo, autorizzato e non, può effettuare delle operazioni maliziose nella rete. L'obiettivo delle due classi di attaccanti è quello di trarre dei benefici personali o semplicemente di apportare un disturbo agli altri nodi della rete. Gli attaccanti interni hanno un impatto maggiore rispetto a quelli esterni

in quanto, essendo *autorizzati* ad accedere alla rete, possono mimetizzarsi facilmente, avere una migliore conoscenza della rete e compiere il proprio lavoro senza esporsi troppo. La conoscenza della rete è indispensabile all'attaccante per capirne la struttura e la configurazione. Per questo motivo un attaccante esterno, agendo da intruso, ha un range di possibili attacchi molto più ristretto e molto meno "potente" rispetto a quello interno.

- ***razionali e maliziosi***
  Gli attaccanti maliziosi non hanno l'obiettivo di trarre dei benefici personali ma di arrecare dei danni agli utenti ostacolando le normali funzionalità della rete e delle sue applicazioni. Questo comporta la non considerazione dei costi, dei mezzi di attacco e la non cura delle conseguenze che tale attacco comporta alla rete. Al contrario quelli razionali agiscono per trarre dei profitti personali dai loro attacchi e quindi per questo motivo sono più prevedibili rispetto ai primi.

4.3.2 Classificazione degli attacchi in base alla natura

Dopo aver individuato le famiglie di attaccanti passiamo alla classificazione dei vari tipi di attacchi. Idealmente è possibile individuare 5 diverse classi di attacchi in base alla loro natura [49]:

- ***Network Attack (NA)***
  E' una delle peggiori tipologie di attacco in quanto mira a colpire l'intera rete causando dei malfunzionamenti alle comunicazioni tra i nodi. L'obiettivo di questi attacchi è quello di limitare o annullare la disponibilità della rete e dei servizi da essa offerti. Fa parte di questa classe il famoso attacco DOS (Denial of Service).

- ***Application Attack (AA)***
  Nelle VANET sono potenzialmente presenti due tipologie di applicazioni: applicazioni per la sicurezza ed applicazioni per servizi aggiuntivi. Questa classe di attacchi è una delle più "cattive" in quanto l'obiettivo dell'attaccante è quello di cambiare il contenuto dei messaggi generati dalle varie applicazioni attive per arrecare un danno ai veicoli, ad esempio per causare degli incidenti. Se l'attacco dovesse colpire le applicazioni di sicurezza allora il risultato potrebbe essere catastrofico poichè l'attaccante potrebbe segnalare agli utenti inconsapevoli uno scenario non veritiero. Ad esempio un attaccante potrebbe comunicare un "via libera" in una situazione di incidente o di ingorgo nei pressi di un cantiere stradale. Alcuni dei messaggi di sicurezza utilizzati nella comunicazione V2V e V2I sono Work Zone, Intersection Collision, Post Crash, Emergency Vehicle Approaching o Breakdown.

- ***Timing Attack (TA)***
  Il principale obiettivo di questa tipologia di attacco è quello di causare un danno nelle comunicazioni tra i nodi della rete attraverso la modifica del time slot per generare un ritardo nella trasmissione dei pacchetti. Una volta

ricevuto il pacchetto, l'attaccante non provvederà alla sua ritrasmissione ma lo memorizzerà e ne varierà il time slot causando quindi dei ritardi nel processo di forwarding. In questo modo i messaggi non arrivando in tempo a destinazione provocheranno un malfunzionamento alle cosiddette "*time critical applications*", ossia a quelle applicazioni orientate alla sicurezza degli utenti, facendo quindi venir meno l'utilità di questa tipologia di rete.

- ***Social Attack (SA)***
  Rientrano in questa categoria gli attacchi sferrati da utenti maliziosi aventi l'obiettivo di infastidire gli altri veicoli durante i loro spostamenti mediante l'utilizzo di messaggi sgradevoli contenenti insulti o frasi equivoche.

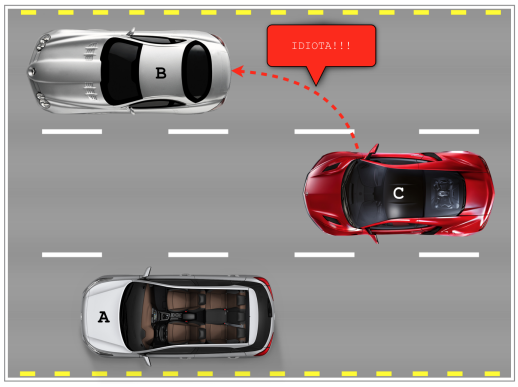
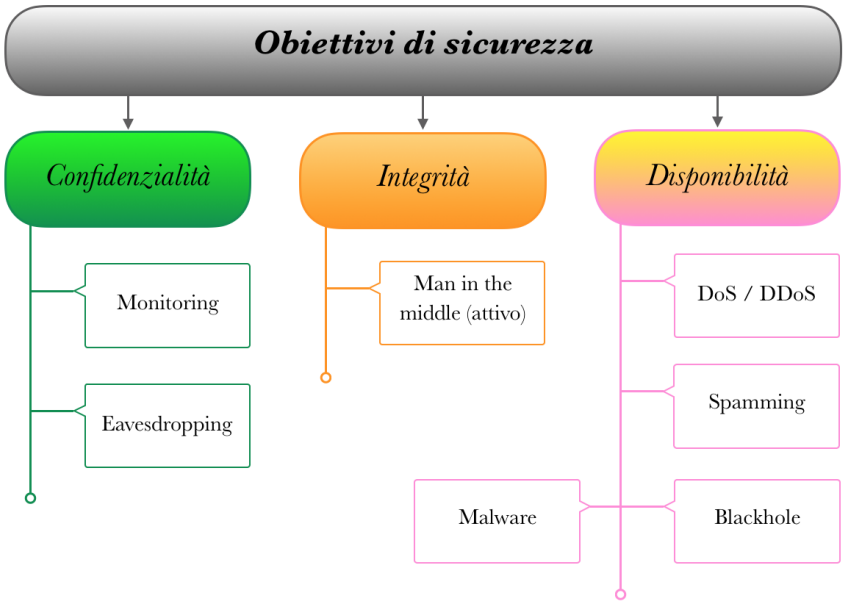


Figura 4.1: Esempio di attacco Social

- ***Monitoring Attack (MA)***
  Gli attacchi appartenenti a questa classe fanno uso di sistemi di monitoring che permettono agli attaccanti di spiare le comunicazioni V2I e V2V per conoscere le informazioni riservate contenute sui beacon, come la posizione o il testo dei messaggi, per avere una migliorare conoscenza degli obiettivi prima di sferrare un attacco.

4.3.3 Classificazione degli attacchi in base agli "obiettivi di sicurezza"

Un metodo alternativo con il quale effettuare un ordinamento dei possibili attacchi è quello proposto in [50] che fa uso dei requisiti di sicurezza che un utente malizioso potrebbe voler infrangere inscenando un attacco. Tra i requisiti di sicurezza mostrati nella parte iniziale del presente capitolo quelli qui utilizzati come unità di misura sono Confidenzialità, Integrità e Disponibilità. Ogni requisito rappresenta per una VANET una proprietà da preservare e proteggere ma volendo essere obiettivi tra le tre la disponibilità è in assoluto la più importante in quanto rappresenta un "elemento" fondamentale per una rete che mira ad offrire dei servizi che hanno l'obiettivo di proteggere le "vite umane". Poiché la descrizione dei possibili attacchi sarà oggetto di discussione della sezione successiva, qua viene solamente mostrata la suddetta classificazione attraverso il seguente grafo:



4.3.4 Classificazione degli attacchi in base alla rilevazione

Gli attacchi possono avere un impatto differente sulla rete e sui nodi in base alla tecnologia ed alle tecniche usate dall'attaccante per compierli. Oltre alla classificazione appena proposta è possibile effettuare un'ulteriore suddivisione degli attacchi basata stavolta sulla loro rilevazione:

- ***Attacco non rilevato dai nodi vittima***
  Un nodo vittima non può rilevare la presenza di un attacco in corso, ad esempio, se isolato o circondato completamente da nodi maliziosi che immettono informazioni fasulle sulla rete. Così facendo il nodo vittima accetterà tutte le informazioni ricevute e potrà *eventualmente* rivalutarne la veridicità solo in seguito quando incontrerà sul tragitto dei nodi onesti.

- ***Attacco rilevato dai nodi vittima***
  Un nodo vittima rileva qualche anomalia sulla rete e quindi capisce che vi è un attacco in corso. Le informazioni fasulle ricevute dai nodi maliziosi vengono trattenute lasciando il nodo in dubbio sulla loro veridicità e verranno rivalutate quando riceverà le stesse informazioni da un numero sufficiente di nodi onesti.

- ***Attacco corretto dai nodi vittima***
  Un nodo vittima individua la presenza di un nodo vicino malizioso e quindi rileva un attacco in corso grazie alla ricezione di informazioni errate completamente differenti da quelle già imparate in precedenza ed apprese da un certo numero di nodi onesti che verranno perciò scartate e non ritrasmesse.

4.3.5 Attacchi: esempi e contromisure

In questa sezione vengono proposti una serie di attacchi con i possibili schemi di prevenzione da mettere in atto per la protezione della rete. Ognuno di questi attacchi

colpisce almeno uno dei requisiti di sicurezza prima indicati. Partiamo considerando un caso classico: l'attacco DOS.

- ***Denial Of Service (DOS)***
  Un attacco DOS può causare gravi malfunzionamenti su ogni tipo di rete. La *disponibilità* è una proprietà molto importante per una rete come la VANET alla quale gli utenti fanno affidamento per la salvaguardia della loro sicurezza stradale. Per questo motivo un attacco DOS è senza dubbio uno dei peggiori attacchi. L'obiettivo principale è fare in modo che gli utenti autorizzati non abbiano la possibilità di accedere ai servizi o alle risorse offerti dalla rete.

  Un possibile schema per la prevenzione di questi attacchi è il modello IP-CHOCK [51] che basa il suo funzionamento sulla verifica delle informazioni di rete dei nodi da parte delle OBU per scovare eventuali indirizzi duplicati e provare a individuare anzitempo possibili situazioni riconducibili ad attacchi DOS.

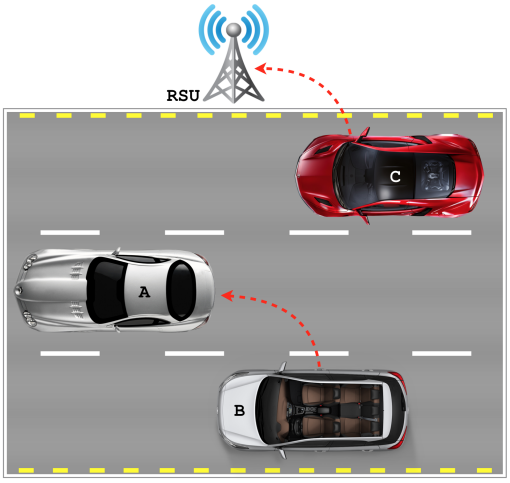
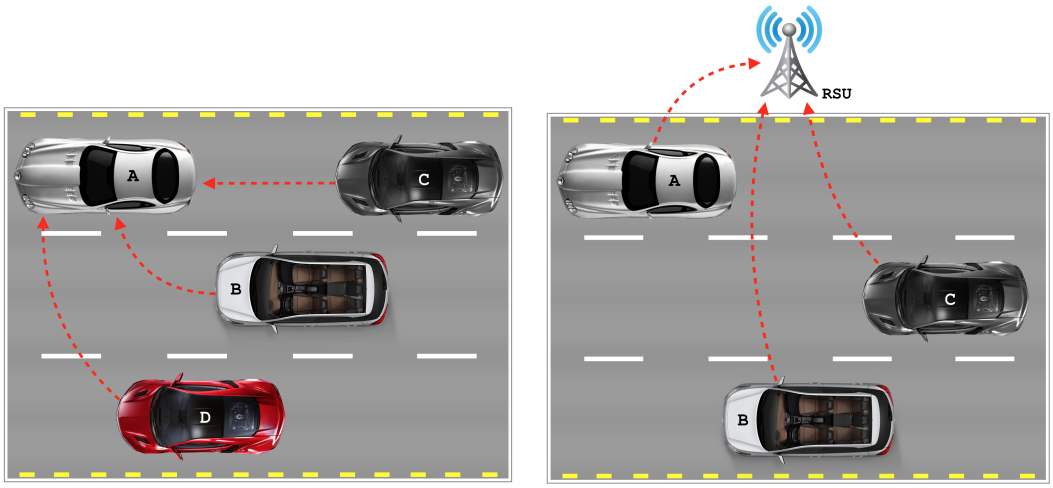


Figura 4.2: Esempio di attacco DoS

- ***Distributed Denial Of Service (DDOS)***
  Un attacco di questo tipo può avere effetti ancora più devastanti rispetto al semplice DOS dal momento che l'attacco è stavolta *distribuito*. In questo caso gli attaccanti (daemon) lanciano gli attacchi da diversi punti della rete per coprire un'estesa area della rete stessa. L'obiettivo è sempre lo stesso: causare un malfunzionamento generale della rete limitando le possibilità di accesso ai nodi autorizzati.

- ***Session Hijacking***
  L'attaccante preleva il *Session Identifier (SID)* assegnato ad ogni sessione e attraverso questo prende il controllo della sessione già instaurata. Le possibili contromisure riguardano l'utilizzo di schemi di cifratura o di generazione random dei SID.

- ***Attacco Replay***
  In questo tipo di attacco, un attaccante ha l'obiettivo di spacciarsi come un



(a) (b)

Figura 4.3: Esempio di attacco DDoS sferrato a un veicolo (a) e all'infrastruttura di rete (b)

utente autorizzato o una RSU attraverso l'utilizzo di messaggi precedentemente ricevuti o catturati che vengono quindi ripetuti in diverse parti della rete infrangendo l'autenticità e la confidenzialità del sistema.

Una soluzione per ovviare ad attacchi di questo tipo potrebbe considerare l'utilizzo di un clock globale per la sincronizzazione di tutti i nodi della rete e l'associazione dei timestamp alle informazioni trasmesse attraverso i beacon.

In letteratura sono presenti una serie di possibili contromisure ed una in particolare è il protocollo ARIADNE, basato sul famoso protocollo di routing DSR, che fa uso della crittografia simmetrica e di due chiavi distinte per ogni verso della comunicazione tra due nodi [52].

- ***GPS Spoofing***
  Questo attacco permette all'attaccante di manipolare il segnale GPS ricevuto all'interno di una certa area della rete in modo arbitrario e di conseguenza di inviare ai nodi informazioni fasulle.

  Diverse contromisure sono state sviluppate ma alcune di esse comportano un aumento dei costi per la costruzione dei dispositivi da installare a bordo dei veicoli e per questo motivo non sono mai state prese in considerazione. La manipolazione del segnale GPS permette però l'espletamento di una serie di attacchi diversi e in [53] vengono mostrate le ultime "scoperte" fatte in merito.

- ***Sybil***
  In questa tipologia di attacco, l'attaccante invia una serie di messaggi con ID diversi con una certa frequenza agli altri nodi per far credere che siano stati generati da veicoli differenti e simulare del traffico sulla rete stradale. L'obiettivo è quello di convincere gli altri veicoli a cambiare il loro tragitto affinché l'attaccante possa trarne dei benefici personali. L'attacco Sybil quindi danneggia la topologia della rete e causa un eccessivo consumo della banda a disposizione.

Le tecniche da mettere in atto per prevenire questi attacchi sono:

- *registrazione dei veicoli* attraverso una corrispondenza uno a uno tra veicolo e id;
- *verifica della posizione dei veicoli* per assicurare che ogni veicolo abbia un solo id ad esso associato.

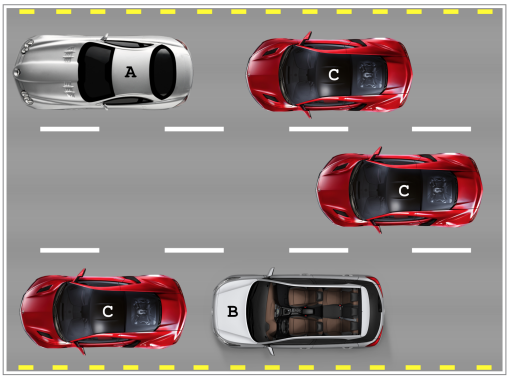


Figura 4.4: Esempio di attacco Sybil

- ***Node Impersonation***
  Ogni nodo in una rete VANET viene identificato da un ID univoco inserito in un apposito campo nei beacon inviati. Mediante questo tipo di attacco, un attaccante finge di essere qualcun altro usando un ID diverso dal suo e cambiando il contenuto dei messaggi ricevuti per generare delle anomalie nella rete. Un possibile scenario potrebbe esser quello in cui un attaccante finge di essere un nodo "buono" variando la propria identità per sfuggire ad un controllo della forze dell'ordine dopo aver causato un incidente. Questi attacchi sono resi possibili attraverso furti di identità o possesso di certificati falsi e potrebbero essere combattuti mediante l'utilizzo di una *Trust Authority (TA)* che consenta di verificare in ogni momento la veridicità dell'identità del nodo e di una *Public Key Infrastrutture (PKI)*.

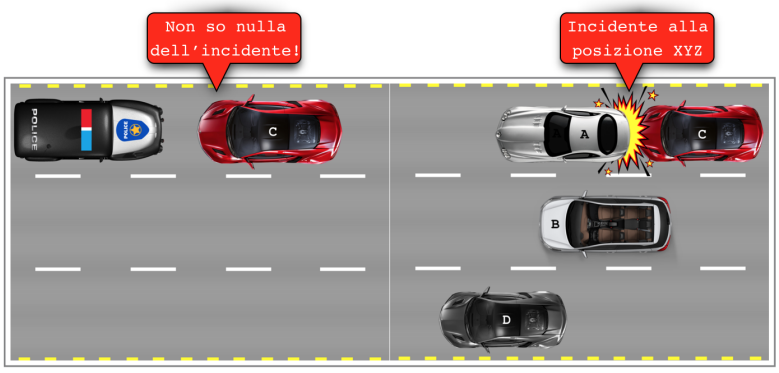


Figura 4.5: Esempio di attacco Node Impersonation

- ***Eavesdropping***
  Questo è un attacco messo in atto dai cosiddetti attaccanti passivi che si limitano semplicemente ad intercettare la comunicazione per prelevare informazioni utili contenute nei messaggi da utilizzare successivamente per scopi precisi. Una possibile contromisura può essere l'utilizzo di tecniche crittografiche per la protezione delle informazioni racchiuse nei messaggi.

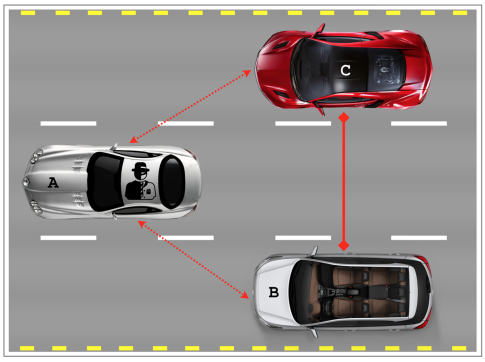


Figura 4.6: Esempio di attacco di Eavesdropping

Oltre alle più interessanti tipologie di attacco fin qui descritte possono essere messi in atto anche gli attacchi più "classici" delle reti come il *black hole*, lo *spamming* oppure il *malware* per limitare le funzionalità della rete e ancora il *man-in-the-middle* o il metodo *forza bruta* per manipolare rispettivamente la comunicazione tra i nodi e l'identità dei veicoli (*VIN*) che prendono parte al processo di comunicazione. Una ricapitolazione delle proprietà dei diversi attacchi e delle eventuali contromisure è indicata nella Tabella 4.2.

Nel corso degli anni la comunità scientifica ha dibattuto molto sulle vulnerabilità delle reti VANET sfruttabili dagli utenti maliziosi per compiere una vasta gamma di possibili attacchi e numerosi modelli per prevenirli sono stati sviluppati. Purtroppo non tutti gli schemi proposti riescono a raggiungere gli scopi prestabiliti in modo efficace anche a causa delle difficoltà nello sviluppo in un contesto reale di queste tecniche. Tra le soluzioni proposte in letteratura [54] e [55] sono le più interessanti per il contrasto agli attacchi come il Node Impersonation, l'Eavesdropping, il Replay ed il Session Hijacking. Questi propongono dei modelli denominati ARAN (*Authenticated Routing for Ad-hoc Network*) e OTC (*One Time Cookie*) che si preoccupano rispettivamente di definire un protocollo di routing sicuro basato su AODV che sfrutta il meccanismo di crittografia a chiave pubblica ed uno schema per la generazione randomica dei cookie basato su HMAC per prevenirne il riutilizzo da parte degli utenti maliziosi. Infine, tecniche per il rilevamento delle intrusioni per le ad-hoc network sono state studiate in [56] e [57].

4.4 Privacy e gestione dell'identità

L'emergente tecnologia delle comunicazioni veicolari (VC) solleva una serie di criticità: privacy e anonimato. Da una corretta gestione di queste proprietà dipende l'effettiva adozione futura di questo sistema di comunicazione.

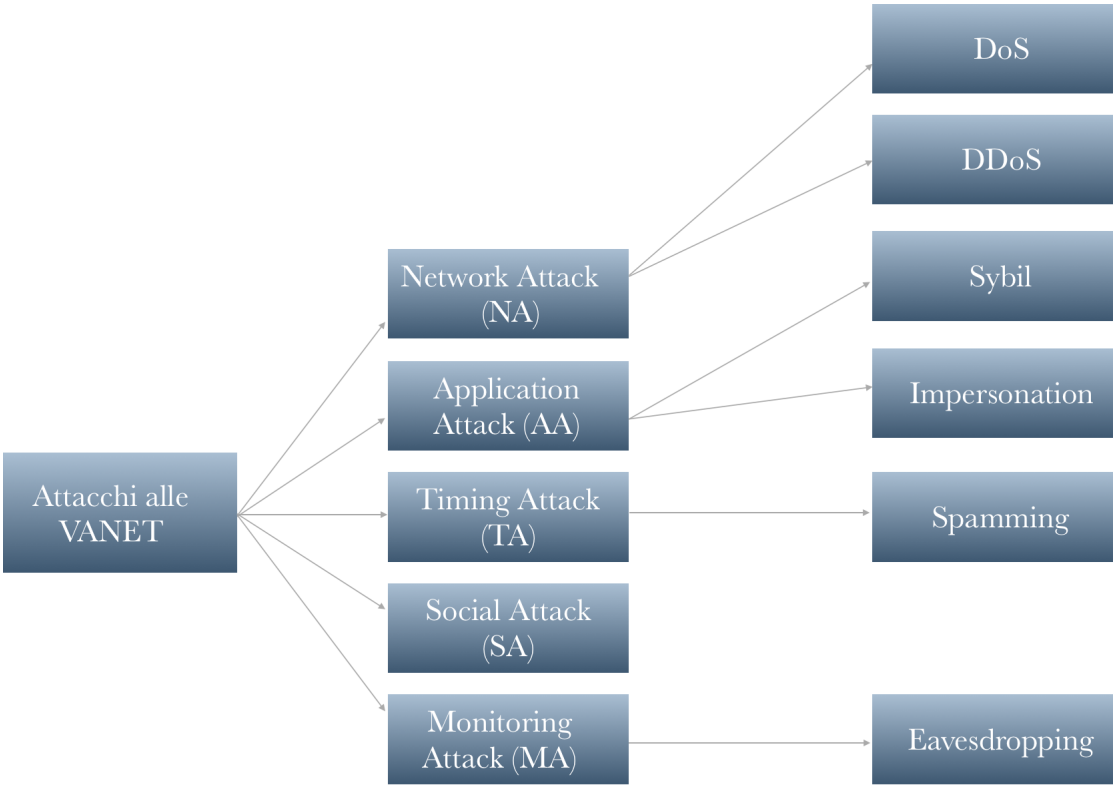


Figura 4.7: Classificazione degli attacchi in base alla natura e relativi esempi

Oggi si sente parlare molto di privacy e dei problemi ad essa connessi a causa della forte espansione avuta nel corso dell'ultimo ventennio dalla rete Internet e dalle sue applicazioni. La privacy di ogni veicolo e dei suoi passeggeri deve essere garantita in ogni suo aspetto, sia nei confronti degli altri veicoli che nei confronti delle autorità.

In Europa il principale progetto di ricerca che si è preoccupato di considerare la privacy e la sicurezza nelle reti veicolari è stato il progetto SeVeCom (*Secure Vehicular Communications*) [58].

4.4.1 Pseudonimi

Nel mondo delle VANET il problema della privacy viene in genere gestito mediante l'uso di *pseudonimi* che rendono molto difficile l'associazione dei veicoli con i messaggi da essi generati in quanto questi identificatori sono usati solo per brevi periodi di tempo. Uno pseudonimo non è altro che una chiave pubblica temporanea associata all'identità reale del nodo e certificata da una delle CA che formano la cosiddetta VPKI (*Vehicular Public Key Infrastructure*). Il nodo dovrà contattare con una certa regolarità la propria CA per farsi assegnare un pool di chiavi pubbliche temporanee (processo di ricarica degli pseudonimi).

In realtà con il termine CA si fa riferimento ad una struttura generica che si compone idealmente di tre entità:

- la **LTCA (*Long Term Certification Authority*)** autentica e rilascia i certificati ai veicoli;
- la **PCA (*Pseudonyms Certification Authority*)** assegna gli pseudonimi ai veicoli che ne fanno richiesta;
- la **RA (*Resolution Authority*)** gestisce il processo di verifica e revoca dei certificati.

Ogni veicolo firma i messaggi generati con la chiave privata corrispondente allo pseudonimo che viene allegato al messaggio stesso. Il nodo che riceverà il messaggio si limiterà a verificarne lo pseudonimo e successivamente il contenuto. Questa procedura assicura integrità, autenticità e non ripudio.

Ogni pseudonimo contiene 4 attributi:

- l'identificativo della terza parte fidata che ha fornito lo pseudonimo (CA);
- il periodo di validità dello pseudonimo;
- la chiave pubblica;
- la firma digitale della CA.

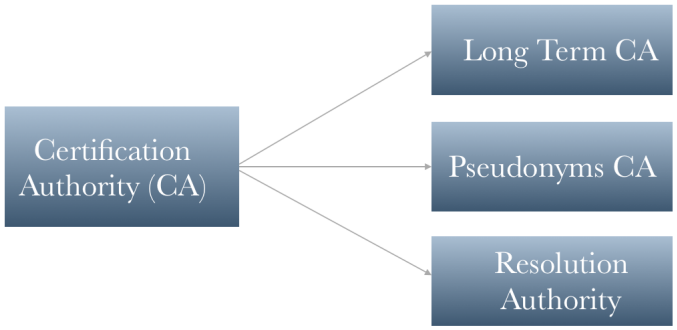


Figura 4.8: Composizione ideale di una Certification Authority

Il continuo cambio di pseudonimi comporta diversi altri problemi come quelli connessi ai protocolli di rete utilizzati. Vi sono infatti delle applicazioni che richiedono delle connessioni di lunga durata tra le parti coinvolte. Spesso queste fanno affidamento a dei sistemi di gestione delle sessioni di rete che effettuano il processo di autenticazione o il controllo del flusso di pacchetti. Una continua variazione degli identificatori causerà quindi una continua riattivazione della sessione con parametri sempre diversi ed apporterà un overhead non indifferente alla rete. Un ulteriore problema causato dalla continua variazione degli pseudonimi è il forte impatto sui protocolli di comunicazione che devono essere sviluppati per consentire una gestione ottimale della trasmissione dei messaggi. Ad esempio il routing geografico fa un forte affidamento agli identificatori dei nodi del vicinato e una continua modifica di questi valori può ostacolarne il corretto funzionamento.

Il punto è: con quale frequenza gestire l'aggiornamento degli identificatori dei nodi? Cambiandoli una volta al giorno potrebbe essere una buona soluzione per evitare lunghe operazioni di tracking ma di contro potrebbe non essere sufficiente per evitare che un investigatore privato o un utente malizioso pedini la vittima durante l'intera giornata. D'altro canto, non sarebbe neppure efficiente un aggiornamento degli pseudonimi a intervalli di qualche secondo in quanto, se da un lato è possibile ottenere una protezione della privacy impeccabile, dall'altro si può andar in contro a continue perdite di pacchetti e problemi di comunicazione poiché non sarà ad esempio possibile per un certo nodo rispondere ad un pacchetto precedentemente inviatogli con un identificatore ormai andato in disuso. Naturalmente non deve essere possibile creare una relazione tra gli identificatori associati di volta in volta ad un certo nodo altrimenti verrà meno l'utilità di questa continua variazione degli pseudonimi.

In [59] vengono valutati diversi scenari relativi all'impatto sulla rete del processo di aggiornamento degli identificatori e analizzati gli effetti e, mediante simulazioni e formule matematiche, viene data una definizione della percentuale dei pacchetti persi a causa della modifica degli pseudonimi valutando l'eventuale influenza della densità dei nodi e della loro velocità. Altri studi in merito hanno proposto una gestione differente e più autonoma delle credenziali nelle VANET, un'architettura basata sulla firma di gruppo in cui sono i nodi stessi a generare di volta in volta gli pseudonimi senza fare affidamento a nessuna entità esterna al fine di ridurre il carico apportato alla rete dal sistema precedentemente esposto di *ricarica degli pseudonimi* [60].

A causa della difficoltà riscontrata nello sviluppo di un sistema di sicurezza efficiente e "leggero", i progettisti di sistema delle VANET devono puntare ad un compromesso tra la tutela della privacy e le prestazioni della rete.

4.5 Tabelle riassuntive

Nel presente paragrafo è possibile trovare delle tabelle riassuntive dei diversi tipi di attaccanti con i relativi obiettivi (Tabella 4.1) e delle principali classi di attacchi conducibili alle reti VANET con obiettivi e possibili contromisure (Tabella 4.2).

Tipologia Attaccante	**Obiettivo**
Attivo	Utente mimetizzato nella rete con l'obiettivo di danneggiare gli utenti autorizzati mediante la manipolazione dei messaggi
Passivo	Utente che agisce "dietro le quinte" effettuando operazioni di sniffing o monitoring dei messaggi scambiati tra gli utenti autorizzati
Interno	Membro interno, quindi autenticato e autorizzato, avente l'obiettivo di causare dei danni mediante attacchi molto efficaci grazie alla conoscenza della rete
Esterno	Membro esterno alla rete che agisce da intruso per compiere degli attacchi che mirano a limitare le funzionalità della rete
Razionale	Entità esterna o interna alla rete che compie un attacco di tipo attivo per trarne dei benefici personali
Malizioso	Entità esterna o interna alla rete avente il solo obiettivo di generare dei disastri sulla rete e sugli utenti della rete

Tabella 4.1: Classificazione delle diverse tipologie di attaccanti

Attacco	Obiettivo	Contromisura
DoS	Limitare la disponibilità della rete per fare in modo che utenti autorizzati non abbiano la possibilità di accedere ai servizi o alle risorse di rete	Non esistono contromisure definitive; una possibile soluzione è IP-CHOCK [51]
DDoS	Come il DoS ma l'effetto è moltiplicato per il numero di daemon che prendono parte all'attacco	Non sono presenti delle contromisure definitive
Sybil	Fornire ai veicoli autorizzati l'illusione della presenza di traffico nei propri dintorni al fine di convincerli al cambio di rotta per trarne dei benefici personali	Registrazione dell'identità dei veicoli presso un'autorità fidata e verifica della posizione dei veicoli mediante radar
Node Impersonation	Impedire l'individuazione del proprio veicolo variando il proprio ID attraverso furti di identità o certificati falsi	Definizione di un'autorità che verifichi la veridicità dei dati dei nodi e procedure di autenticazione
Eavesdropping	Intercettare la comunicazione per prelevare le informazioni sensibili contenute nei messaggi	Utilizzo di schemi di crittografia per la protezione delle informazioni racchiuse nei messaggi
Replay	Ripetere in diverse parti della rete i messaggi catturati per disorientare le autorità ed impedire l'individuazione del nodo malizioso	Utilizzo di un clock per la sincronizzazione dei nodi e di timestamp da inserire negli header dei pacchetti
Session Hijacking	Prendere il controllo della sessione instaurata per manipolarla a piacimento	Adozione di schemi di cifratura o di generazione random dei Session Indentifier

Attacco	**Obiettivo**	**Contromisura**
GPS Spoofing	Manipolare il segnale GPS e inviarlo per trasmettere delle informazioni fasulle sulla posizione	Criptazione del segnale del sistema di rilevazione
Black Hole	Controllare il traffico di un'intera zona della rete reindirizzandolo verso uno specifico nodo	Implementazione di uno schema di routing sicuro
Spamming	Inondare la rete con messaggi di spam per limitarne le funzionalità ed aumentare la latenza della trasmissione dei pacchetti	Utilizzo di sistemi antispam
Malware	Immettere una qualche forma di virus sulla rete per ostacolarne il normale svolgimento delle funzionalità	Utilizzo di antivirus
Man-In-The-Middle	Manipolare la comunicazione tra i nodi mediante l'immissione o la modifica dei messaggi scambiati	Uso di chiavi crittografiche firmate da una terza parte fidata (CA)
Forza bruta	Azione effettuata sui VIN dei veicoli prelevandone e compromettendone l'identità	Adozione di tecniche di autenticazione

Tabella 4.2: Classificazione dei principali tipi di attacco e relative contromisure

Capitolo 5

Progettazione

La VANET, quando implementata, sarà certamente una delle reti più grandi per estensione e quantità di informazioni scambiate. Come si è potuto vedere dai capitoli precedenti questa rete può soffrire di una serie di problemi, come la privacy dei nodi, che hanno senza dubbio ostacolato e reso più arduo l'iter di sviluppo per la messa in opera finale. In effetti è da molto tempo ormai che la comunità scientifica dibatte sui limiti delle VANET e molte delle proposte avanzate, seppur eccezionali per modalità di esecuzione e di raggiungimento degli scopi, sono risultate vane a causa delle limitate capacità di calcolo dei componenti che ne definiscono l'architettura. Tuttavia questi studi sono particolarmente importanti in quanto possono rappresentare il punto di partenza per lo sviluppo di nuove e più efficienti soluzioni.

L'applicazione sviluppata in questo lavoro di tesi ha l'obiettivo di analizzare la sicurezza delle reti VANET mediante lo studio di tecniche utili per la mitigazione degli attacchi e per la protezione dei servizi ad oggi disponibili. Come tutti i casi di studio effettuati in questo ambito il punto di partenza è la simulazione dello scenario reale sul quale andare poi ad attuare le strategie sviluppate per lo studio di uno specifico problema.

Nel presente capitolo è mostrata inizialmente l'architettura di alto livello della soluzione sviluppata e successivamente messo in risalto il cosiddetto attack model, ovvero quel modello che evidenzia le criticità e quindi i possibili scenari fruibili dagli attaccanti per far venir meno almeno uno dei requisiti di sicurezza richiesti per la "protezione" di una rete VANET.

5.1 Simulazione con SUMO

Lo sviluppo dell'applicazione parte dalla simulazione di uno scenario di traffico mediante l'uso del simulatore SUMO nella Release 0.30.0, descritto ampiamente nel Paragrafo 2.9 del Capitolo 2, che permette di ottenere i dati (tracce) dai quali partire per simulare successivamente la comunicazione e lo scambio di beacon tra i veicoli.

Il file XML che descrive la rete stradale e che deve essere dato in pasto a SUMO può essere editato manualmente dall'utente o in alternativa può essere generato

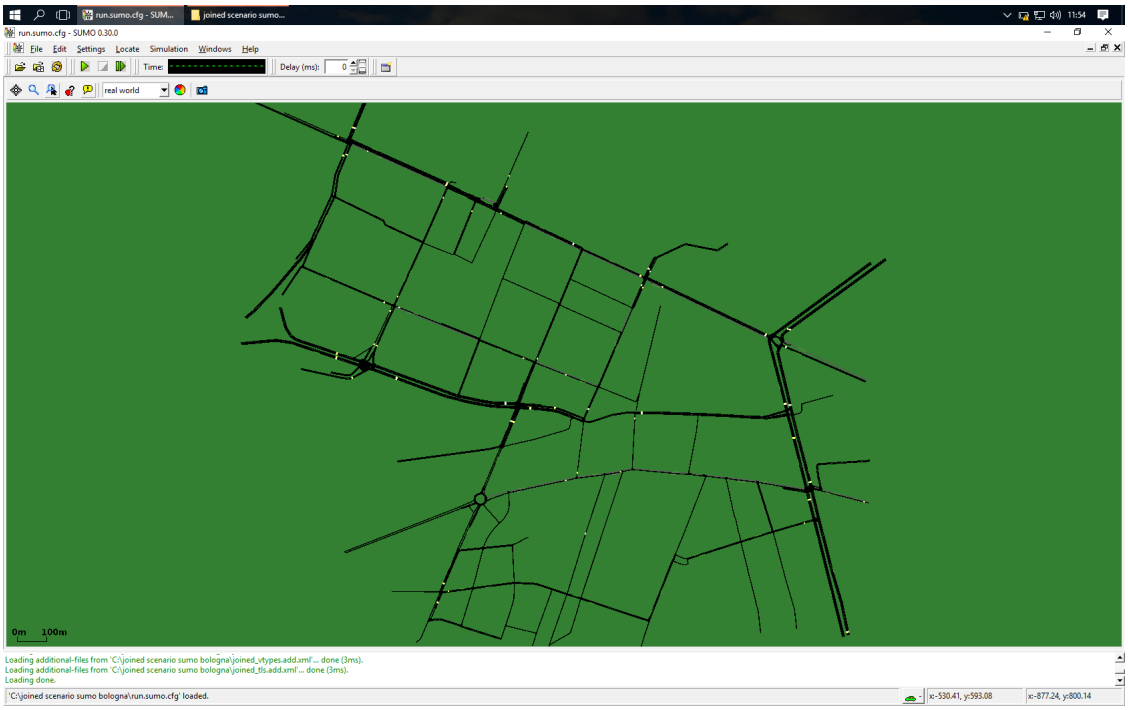


Figura 5.1: Scenario simulato con SUMO

ed esportato da altri tool che, tra le altre cose, permettono di importare in SUMO mappe di città realmente esistenti. Opzionalmente, mediante l'applicazione POLYCONVERT [61], è possibile definire dei file aggiuntivi che descrivono minuziosamente l'ambiente cittadino in modo da condurre le simulazioni in un contesto più realistico possibile e non frutto della fantasia dell'utente.

All'interno dello scenario simulato i veicoli devono spostarsi seguendo delle regole o meglio delle rotte predefinite. Il metodo più semplice per la definizione di queste rotte è quello di modificare file già esistenti a piacimento oppure di utilizzare l'applicazione DFROUTER [62] che fa uso del conteggio dei veicoli (ad esempio attraverso i cosiddetti cicli o sensori induttivi) per il calcolo dei percorsi dei veicoli e ha inoltre la capacità di analizzare il flusso del traffico. In ogni caso, all'interno del package di SUMO è possibile trovare diversi tutorial completi con i relativi file XML da utilizzare come punto di partenza per la creazione di un contesto di studio virtuale personalizzato.

In questo specifico caso si è fatto uso di uno scenario di traffico realistico predefinito nato dal progetto di ricerca iTETRIS [63] che rappresenta in scala una parte della città di Bologna e che risulta essere una scelta valida per lo scopo del presente lavoro di tesi. Tra le possibili scelte vi è anche lo scenario chiamato TAPAS Cologne che descrive però in modo più ampio il traffico nella città tedesca di Colonia [64]. Questo sistema calcola la mobilità di un'area di popolazione sulla base delle informazioni relative alle abitudini di mobilità dei cittadini tedeschi. I package di questi due preset includono la mappa della città, le cui informazioni sono racchiuse in una struttura in formato XML, e tutte le specifiche, sempre sotto forma di file XML, che descrivono in ogni minimo aspetto lo scenario cittadino considerato. La qualità dei dati in input è di fondamentale importanza per l'ottenimento di risultati realistici.

Lo scenario di partenza può essere definito solo dopo aver editato un file di configurazione che contenga tutte le informazioni citate in precedenza. Inoltre esso dovrà contenere la durata della simulazione e la frequenza di campionamento degli spostamenti dei veicoli che permetterà poi di ottenere in uscita un file contenente tutte le tracce per ogni `TIME_STEP` considerato. Una volta definito, questo file potrà essere importato da SUMO per la visualizzazione tramite la sua interfaccia grafica della mappa della città e successivamente eseguito per l'avvio della simulazione (Figura 5.1). In realtà non esiste un solo output in SUMO ma l'utente può decidere a piacimento, a seconda delle situazioni e degli scopi, quale sia tra quelle disponibili la migliore rappresentazione delle informazioni [65].

Il nostro obiettivo è quello di accedere alle tracce dei veicoli in movimento sull'asse stradale al fine di studiare i possibili attacchi che un utente malintenzionato può mettere in atto durante la comunicazione V2V (Vehicle-to-Vehicle) e V2I (Vehicle-to-Infrastructure).

La simulazione avviata con SUMO da linea di comando produce un file XML di dimensione variabile che, attraverso una serie di attributi, descrive le proprietà dei veicoli presenti. In particolare, per ogni veicolo, le principali grandezze sono:

- id;
- velocità;
- corsia;
- coordinate spaziali.

Il formato di uscita scelto è FCDOutput (*Floating Car Data Output*). Questo consiste in un file di testo in formato XML avente per ogni singolo istante di campionamento (`TIME_STEP`) la seguente struttura:

```
<timestep time="TIME_STEP">

    <vehicle slope="VEHICLE_SLOPE" lane="VEHICLE_LANE"
             pos="VEHICLE_POS"  speed="VEHICLE_SPEED"
             type="VEHICLE_TYPE" angle="VEHICLE_ANGLE"
             y="VEHICLE_POS_Y"    x="VEHICLE_POS_X"
            id="VEHICLE_ID"
    />

    . . . altri veicoli. . .

</timestep>
```

Gli attributi che vengono descritti per ogni veicolo istante per istante sono sommariamente raccolti e spiegati nella Tabella 5.1.

Attributo	Significato
Slope	Pendenza della corsia misurata in *gradi*
Lane	Nome della corsia in cui si trova il veicolo
Pos	Posizione del veicolo su una specifica corsia
Speed	Velocità attuale del veicolo misurata in m/s
Type	Nome del tipo di veicolo
Angle	Pendenza del veicolo misurata in *gradi*
x	Coordinata assoluta x del veicolo
y	Coordinata assoluta y del veicolo
ID	Identificativo del veicolo
Time	Istante di tempo considerato misurato in s

Tabella 5.1: Sommario degli attributi dei veicoli descritti dall'outout FCD di SUMO

Questa particolare tipologia di output appena presa in esame contiene in sostanza le informazioni relative alla posizione e alla velocità dei veicoli. Le informazioni sulla posizione possono essere, a scelta dell'utente, trasformate in coordinate geografiche rappresentanti la latitudine e la longitudine dei veicoli mediante l'utilizzo del flag `fcd-output.geo` da riga di comando al lancio della simulazione. L'uscita si comporta in qualche modo come un dispositivo GPS ad alta frequenza super-preciso per ogni dato veicolo. Essa potrà poi essere processata successivamente con altri tool come TraceExporter [66] per adattare la frequenza, l'accuratezza e il formato finale dei dati prelevati.

5.2 Tracce: estrazione e salvataggio

Nel paragrafo precedente è stata presentata sinteticamente la modalità di esecuzione di una simulazione in SUMO e il file di output FCD (*Floating Car Data*) che ho deciso di utilizzare per le finalità di studio del presente lavoro di tesi. Purtroppo gli attributi contenuti nel suddetto file non sono selezionabili dall'utente e le tracce descritte mediante il formato visto in precedenza non sono esportabili su un database per la successiva elaborazione. La soluzione ideale sarebbe la traduzione del file di partenza in un altro file con formato CSV (*Comma Separated Value*) da importare successivamente su una base dati opportunamente definita.

La presenza del database è fondamentale per il nostro scopo in quanto lì è possibile memorizzare tutte le informazioni ottenute dalla simulazione per effettuare al momento opportuno interrogazioni ed elaborazioni dei dati.

Nel panorama odierno sono presenti due principali soluzioni relative ad altrettante "tipologie" di database da prendere in considerazione per ogni specifico scopo. Si tratta dei due più importanti RDBMS (*Relational DataBase Management System*): PostGRES e MySQL. Fin dal momento della loro apparizione sul mercato si è discusso molto su quale tra i due fosse il sistema più efficiente, anche se parlando di fama MySQL è senz'altro il più conosciuto tra i due.

PostgreSQL è *ACID compliant* e, a differenza di MySQL, segue lo standard SQL in modo più "stretto". L'acronimo ACID sta ad indicare quattro importanti proprietà: *Atomicity, Consistency, Isolation e Durability*; ciò vuol dire che in PostgreSQL le query proteggeranno l'integrità dei dati e ritorneranno lo stesso output senza errori grazie al concetto di atomicità delle operazioni. Infatti, uno dei principali limiti di MySQL è proprio quello dell'affidabilità. Il modo con cui sono trattate funzionalità quali transazioni o riferimenti lo rendono meno affidabile rispetto al principale avversario. Di contro, MySQL offre agli utilizzatori una maggiore sicurezza, intesa come protezione per l'accesso e/o l'utilizzo dei dati. Per concludere questo mini confronto possiamo quindi affermare che nei casi in cui sia importante l'integrità dei dati è bene prendere in considerazione PostgreSQL mentre nei casi in cui siano importanti le prestazioni e la sicurezza, come in questo caso, ha invece senso considerare MySQL.

Il file in uscita dal simulatore contenente le tracce, come detto più volte, ha estensione *.xml* e la sua traduzione nel formato CSV può avvenire mediante l'utilizzo di un tool apposito o di un parser. In precedenza si è fatto riferimento al tool TraceExporter che, seppure abbia la capacità di creare un file con estensione *.csv* per come richiesto in questo caso, sembrerebbe che in realtà non generi un file idoneo per la successiva importazione su una base di dati delle informazioni ivi rappresentate. A ragion di ciò, cercando tra i numerosi tool presenti, ho trovato utile *xml2csv* [67] che potrebbe addirittura risultare più efficiente rispetto al precedente. Questo strumento infatti si limita a convertire ogni tipo di file generato da SUMO in un altro file nella rappresentazione Comma Separated Value avente per ogni veicolo nella simulazione gli stessi attributi del file XML originario già visto nel precedente paragrafo e rappresentati nella Tabella 5.1.

5.3 Proprietà tabella e importazione delle tracce

La base di dati conterrà una tabella chiamata *Traces* nella quale verranno salvate le tracce generate da SUMO e tradotte dal tool *xml2csv*. Poiché è noto che la dimensione del file aumenta in maniera smisurata al crescere della durata e della complessità della simulazione, bisognerà prendere in seria considerazione l'utilizzo di script che automatizzino il processo e di meccanismi di pacchettizzazione e/o parallelizzazione che mirino a migliorare le prestazioni di questa fase iniziale dello studio.

La tabella Traces, mostrata in Figura 5.2, conterrà i seguenti campi:

	timestep	vehicle_angle	vehicle_id	vehicle_lane	vehicle_pos	vehicle_slope	vehicle_speed	vehicle_type	vehicle_x	vehicle_y	fake_trace	trace_id
Modifica Copia Elimina	0.3	233.59	Silvani_11_	a203[0]_0	0.15	0	0.74	passenger1	2112.41	1337.83	0	196654
Modifica Copia Elimina	0.3	53.39	XXI_Aprile_	a8_0	0.15	0	0.76	passenger3	586.8	59.92	0	196656
Modifica Copia Elimina	0.3	10.07	Turati_10_0	a195_0	0.16	0	0.8	passenger2b	1332.57	1.02	0	196655
Modifica Copia Elimina	0.3	202.75	bus_12_0	b3[0]_1	0.15	0	0.74	bus	430.43	1916.65	0	196657
Modifica Copia Elimina	0.3	294.77	bus_1_0	a55_0	0.15	0	0.75	bus	2100.82	942.25	0	196658
Modifica Copia Elimina	0.3	33.7	bus_3_0	b13[0]_0	0.15	0	0.73	bus	139.15	1226.2	0	196660
Modifica Copia Elimina	0.3	113.79	bus_2_0	b6_1	0.15	0	0.73	bus	21.22	1888.99	0	196659
Modifica Copia Elimina	0.3	84.69	Gandhi_40_0	b1[0]_2	0.16	0	0.77	attacker4	-0.15	1067.59	1	7
Modifica Copia Elimina	0.3	353.9	Audinot_7_0	a131_0	0.14	0	0.7	attacker5	1865.2	105.64	1	6
Modifica Copia Elimina	0.3	102.4	Certosa_9_0	b22[0]_1	0.16	0	0.78	passenger3	200.92	980.94	0	196648
Modifica Copia Elimina	0.3	236.05	Malvasia_70	b52_0	0.17	0	0.84	passenger2a	1633.11	1402.22	0	196650
Modifica Copia Elimina	0.3	68.91	Costa_12_0	a78[0]_0	0.15	0	0.75	passenger1	350.28	360.74	0	196649
Modifica Copia Elimina	0.4	233.59	Silvani_11_	a203[0]_0	0.25	0	0.96	passenger1	2112.33	1337.77	0	196668
Modifica Copia Elimina	0.4	113.79	Borgo_100_0	b6_0	0.23	0	0.93	passenger4	19.95	1885.95	0	196661
Modifica Copia Elimina	0.4	102.4	Certosa_9_0	b22[0]_1	0.26	0	1.01	passenger3	201.02	980.91	0	196662
Modifica Copia Elimina	0.4	236.05	Malvasia_70	b52_0	0.28	0	1.12	passenger2a	1633.01	1402.16	0	196664
Modifica Copia Elimina	0.4	68.91	Costa_12_0	a78[0]_0	0.24	0	0.96	passenger1	350.37	360.78	0	196663
Modifica Copia Elimina	0.4	348	Pepoli_3_0	a210_2	0.27	0	1.06	ignoring2b	2035.48	84.64	0	196665
Modifica Copia Elimina	0.4	10.07	Turati_10_0	a195_0	0.26	0	1.05	passenger2b	1332.59	1.12	0	196669
Modifica Copia Elimina	0.4	53.39	XXI_Aprile_	a8_0	0.25	0	0.98	passenger3	586.88	59.98	0	196670
Modifica Copia Elimina	0.4	202.75	bus_12_0	b3[0]_1	0.25	0	0.97	bus	430.4	1916.56	0	196671
Modifica Copia Elimina	0.4	84.69	Gandhi_40_0	b1[0]_2	0.26	0	1	attacker4	-0.05	1067.6	1	9
Modifica Copia Elimina	0.4	353.9	Audinot_7_0	a131_0	0.23	0	0.93	attacker5	1865.19	105.73	1	8
Modifica Copia Elimina	0.4	51.2	Pertini_20_	b63[0]_1	0.26	0	1.03	passenger1	14.71	1150.76	0	196666
Modifica Copia Elimina	0.4	202.74	Prati_Capra	b3[0]_0	0.26	0	1.02	passenger3	427.34	1917.83	0	196667

Figura 5.2: Screenshot tabella delle tracce nello scenario di Bologna

- trace_id (chiave primaria);
- timestep_time;
- vehicle_angle;
- vehicle_id;
- vehicle_lane;
- vehicle_pos;
- vehicle_slope;
- vehicle_speed;
- vehicle_type;
- vehicle_x;
- vehicle_y.

Tra tutti gli attributi quelli più rilevanti per il mio caso di studio sono la posizione del veicolo, rappresentata dalle coordinate x e y, e l'id. Tuttavia ad un certo punto dovrò considerare la presenza di tracce fasulle all'interno della tabella per studiare la mitigazione di alcune tipologie di attacco. Per questo motivo agli attributi appena listati trovo indispensabile aggiungerne un altro chiamato *fake_trace* che, a seconda che abbia valore 0 o 1, mi consentirà di riconoscere se quella riga fa riferimento ad una traccia reale o manipolata. Sarà considerata successivamente la modalità di creazione ed inserimento di queste tracce false all'interno della base di dati.

La definizione di un semplice script per l'automatizzazione dell'intero processo di creazione-traduzione-importazione mi permette con un semplice clic del mouse

di avviare la procedura e trovare dopo qualche minuto le tracce dei veicoli generate dal simulatore di traffico direttamente sulla tabella della base di dati pronte per la successiva elaborazione. Il suddetto script sarà poi ripreso, mostrato e spiegato più attentamente nell'Appendice A.

5.4 Creazione delle tracce “fake”

Oltre alle tracce “buone” generate dal simulatore di traffico sullo scenario bolognese, per lo scopo del presente lavoro di tesi trovo indispensabile considerare la creazione delle cosiddette “tracce fake”, cioè quelle tracce che simulano la presenza di nodi maliziosi all'interno della rete per l'intero arco della simulazione. Queste sono particolarmente importanti per lo studio della mitigazione di eventuali attacchi conducibili alle reti VANET da utenti malintenzionati. La tabella che dovrà ospitare le tracce sulla base di dati (Figura 5.2) contiene un opportuno campo denominato “fake_trace” che quando settato al valore 1 identifica una traccia malevola.

La generazione di questa tipologia di tracce non è stata fin qui considerata e per questo motivo bisogna adesso definire una strategia che consenta di farlo. Una soluzione potrebbe essere quella di sviluppare un'applicazione che gestisca la sola creazione delle tracce che saranno poi scritte su di un file in formato CSV oppure caricate direttamente sulla tabella tramite opportuni metodi implementati all'interno dell'applicazione stessa. Un'altra soluzione consisterebbe nella definizione di una seconda simulazione in SUMO sullo stesso scenario che si preoccupi esclusivamente della generazione delle tracce fasulle. Entrambe le soluzioni proposte rappresentano delle valide alternative per l'ottenimento delle tracce fake anche se in termini prestazionali non è possibile dire a priori quale tra le due sia migliore. Nella fattispecie la scelta è ricaduta su quest'ultima strategia che, rispetto alla prima, sembra essere più efficace ed immediata. Infatti la definizione di un'applicazione ad hoc richiederebbe, in buona parte, l'emulazione del lavoro che viene già ottimamente svolto dal simulatore di traffico SUMO. Per fare ciò è stato sufficiente modificare in modo opportuno i file utilizzati nella simulazione precedente. In particolare ho dovuto modificare il file con estensione *.rou.xml* che definisce le rotte seguite dai veicoli attaccanti durante l'arco della simulazione e il file con estensione *.add.xml* che descrive le proprietà dei veicoli maligni, denominati “*attacker*”, che prendono parte alla simulazione.

Una volta definita la strategia per la creazione di queste tracce basta effettuare la medesima procedura seguita nel caso precedente in cui ho gestito il caricamento delle tracce “buone” sul database.

Tutti i comandi da iniettare sulla riga di comando per l'ottenimento di una tabella completa con entrambe le tipologie di tracce sono quindi state raccolte dentro un unico script, consultabile nelle pagine finali di questo documento, all'interno del quale è stato anche inserito un comando che assegna a tutte le tracce fake il valore intero 1 nel relativo campo fake_trace della tabella che le contiene.

Al termine dell'intero iter descritto nei primi 4 paragrafi di questo capitolo sarà possibile trovare all'interno della base di dati una tabella completa con tutte le tracce già pronte per il successivo step nel quale verrà poi eseguita l'elaborazione e la simulazione dello scambio di messaggi tra i veicoli presenti.

	beacon_id	id_src	id_dst	payload	timestamp	trace_id
Modifica Copia Elimina	375	Pertini_20_	Gandhi_40_0	[from=Pertini_20__to=Gandhi_40_0_trace_id=196848]	1.6	196848
Modifica Copia Elimina	376	Pertini_20_	Certosa_9_0	[from=Pertini_20__to=Certosa_9_0_trace_id=196848]	1.6	196848
Modifica Copia Elimina	377	Pertini_20_	Gandhi_50_1	[from=Pertini_20__to=Gandhi_50_1_trace_id=196848]	1.6	196848
Modifica Copia Elimina	378	Pertini_20_	bus_3_0	[from=Pertini_20__to=bus_3_0_trace_id=196848]	1.6	196848
Modifica Copia Elimina	379	Prati_Capra	bus_12_0	[from=Prati_Capra_to=bus_12_0_trace_id=196849]	1.6	196849
Modifica Copia Elimina	380	bus_12_0	Prati_Capra	[from=bus_12_0_to=Prati_Capra_trace_id=196853]	1.6	196853
Modifica Copia Elimina	381	bus_2_0	Borgo_100_0	[from=bus_2_0_to=Borgo_100_0_trace_id=196855]	1.6	196855
Modifica Copia Elimina	382	bus_3_0	Gandhi_40_0	[from=bus_3_0_to=Gandhi_40_0_trace_id=196856]	1.6	196856
Modifica Copia Elimina	383	bus_3_0	Certosa_9_0	[from=bus_3_0_to=Certosa_9_0_trace_id=196856]	1.6	196856
Modifica Copia Elimina	384	bus_3_0	Gandhi_50_1	[from=bus_3_0_to=Gandhi_50_1_trace_id=196856]	1.6	196856
Modifica Copia Elimina	385	bus_3_0	Pertini_20_	[from=bus_3_0_to=Pertini_20__trace_id=196856]	1.6	196856
Modifica Copia Elimina	386	Audinot_7_0	Pepoli_3_0	[from=Audinot_7_0_to=Pepoli_3_0_trace_id=34]	1.7	34
Modifica Copia Elimina	387	Audinot_7_0	Pepoli_9_1	[from=Audinot_7_0_to=Pepoli_9_1_trace_id=34]	1.7	34
Modifica Copia Elimina	388	Gandhi_40_0	Certosa_9_0	[from=Gandhi_40_0_to=Certosa_9_0_trace_id=35]	1.7	35
Modifica Copia Elimina	389	Gandhi_40_0	Gandhi_50_1	[from=Gandhi_40_0_to=Gandhi_50_1_trace_id=35]	1.7	35
Modifica Copia Elimina	390	Gandhi_40_0	Pertini_20_	[from=Gandhi_40_0_to=Pertini_20__trace_id=35]	1.7	35
Modifica Copia Elimina	391	Gandhi_40_0	bus_3_0	[from=Gandhi_40_0_to=bus_3_0_trace_id=35]	1.7	35
Modifica Copia Elimina	392	Borgo_100_0	bus_2_0	[from=Borgo_100_0_to=bus_2_0_trace_id=196857]	1.7	196857
Modifica Copia Elimina	393	Certosa_9_0	Gandhi_40_0	[from=Certosa_9_0_to=Gandhi_40_0_trace_id=196858]	1.7	196858
Modifica Copia Elimina	394	Certosa_9_0	Gandhi_50_1	[from=Certosa_9_0_to=Gandhi_50_1_trace_id=196858]	1.7	196858
Modifica Copia Elimina	395	Certosa_9_0	Pertini_20_	[from=Certosa_9_0_to=Pertini_20__trace_id=196858]	1.7	196858
Modifica Copia Elimina	396	Certosa_9_0	bus_3_0	[from=Certosa_9_0_to=bus_3_0_trace_id=196858]	1.7	196858
Modifica Copia Elimina	397	Gandhi_50_1	Gandhi_40_0	[from=Gandhi_50_1_to=Gandhi_40_0_trace_id=196860]	1.7	196860
Modifica Copia Elimina	398	Gandhi_50_1	Certosa_9_0	[from=Gandhi_50_1_to=Certosa_9_0_trace_id=196860]	1.7	196860
Modifica Copia Elimina	399	Gandhi_50_1	Pertini_20_	[from=Gandhi_50_1_to=Pertini_20__trace_id=196860]	1.7	196860

Figura 5.3: Screenshot tabella dei beacon nello scenario di Bologna

5.5 Beacon: attributi e tabella

Il Capitolo 3 è stato interamente dedicato alla descrizione di ogni singola proprietà di questo importantissimo "componente" che adesso viene ripreso per considerarne la composizione interna ed il salvataggio su una tabella della base di dati che passo dopo passo sta prendendo forma.

Idealmente un beacon dovrebbe possedere alcuni attributi che lo caratterizzano come:

- id;
- id del veicolo che lo ha generato (sorgente);
- id del veicolo al quale è destinato (destinatario);
- posizione del veicolo sorgente;
- velocità del veicolo sorgente;
- contenuto o payload;
- timestamp.

Questi appena elencati saranno alcuni dei campi che andranno a comporre la tabella *Beacons* - mostrata in Figura 5.3 - che ospiterà i messaggi generati e scambiati da tutti i veicoli durante la simulazione eseguita attraverso un'applicazione sviluppata appositamente in Java, chiamata SBX (*SimulBeaconXchange*), a partire

dalle tracce presenti sulla tabella *Traces* delle quali viene salvato il relativo id per avere una relazione diretta traccia-beacon per le future elaborazioni.

Ogni veicolo ha la facoltà di generare un beacon con una certa frequenza (ogni 100ms in questo caso) ed inviarlo ai veicoli che rientrano nell'area circostante definita da una circonferenza avente come centro il veicolo stesso (sorgente) e come raggio il range di trasmissione del segnale predefinito (300 metri in questo caso indipendentemente da eventuali ostacoli). I destinatari dei messaggi saranno quindi tutti coloro che, attraverso opportuni calcoli eseguiti sulla base dei dati relativi alla posizione del sorgente, risiederanno all'interno di questa circonferenza così definita.

Nella Figura 5.4 si può vedere come ognuno dei quattro veicoli, contrassegnati dalle lettere A-B-C-D, copre una certa area, rappresentata da circonferenze aventi un certo raggio dipendente dal range di trasmissione prescelto, che rappresenta la zona dentro la quale devono risiedere fisicamente i veicoli ai quali saranno poi trasmessi periodicamente i beacon sul canale di comunicazione DSRC dedicato.

La definizione del range di trasmissione è uno di quegli argomenti su cui la comunità scientifica ha speso molto nel tentativo di individuare un valore standard che rendesse il processo di beaconing efficiente ed esente da possibili disturbi derivanti dalle collisioni che potrebbero di fatto intaccarne la funzionalità e le prestazioni. Differenti schemi sono stati proposti per controllare la potenza di trasmissione dei beacon, il tasso di trasmissione o la Contention Window a livello MAC, o una combinazione di queste con la definizione di schemi ibridi [13], al fine di ottenere una certa qualità del processo di beaconing nelle VANET.

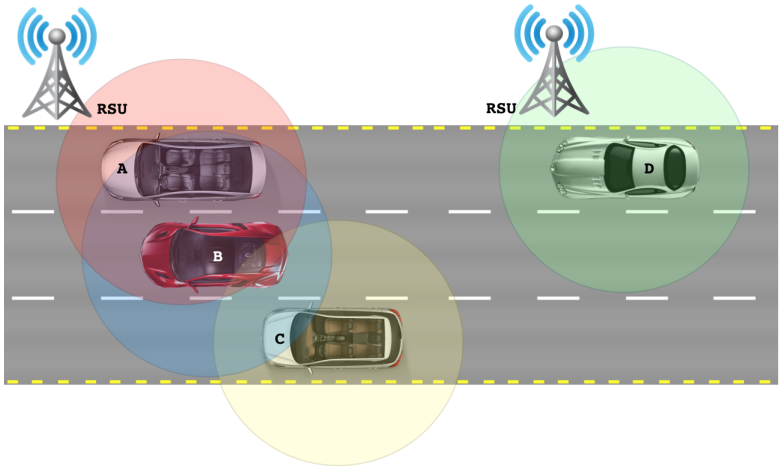


Figura 5.4: Esempio di range di trasmissione

5.6 Creazione e trasmissione dei beacon

Nel paragrafo precedente è stato fatto un accenno all'applicazione *SBX* che ho sviluppato per simulare la comunicazione inter-veicolare tra nodi limitrofi. Il punto di partenza sono le tracce presenti sulla tabella *Traces* che, una volta estratte mediante opportune query, vengono elaborate per simulare lo scambio di beacon. Ogni traccia è riferita ad un certo istante temporale nel quale è stata catturata (`TIME_STEP`) e ad

un certo veicolo del quale vengono misurate alcune grandezze a partire dall'istante iniziale t_0 di lancio della simulazione. E' bene precisare però che non tutti i veicoli entrano in scena all'inizio della simulazione ma possono comparire e scomparire in diversi momenti in relazione alle rotte e alle velocità degli spostamenti predefiniti. La simulazione infatti terminerà non appena tutti i veicoli avranno lasciato lo scenario stradale virtuale utilizzato oppure allo scadere di un timer che, quando impostato, ne indica la durata.

Le unità di misura standard usate da SUMO durante la simulazione sono il *secondo* per il `TIME_STEP` ed il $^m/_s$ (metro al secondo) per la velocità dei veicoli. Questo sta a significare che la tracciatura dei veicoli avverrà secondo per secondo. Purtroppo questa rappresentazione non è adatta al mio scopo poiché tornerebbe più utile un filtraggio ancora più fine misurato in frazioni di secondo che mi consenta di avere da un lato uno scotto da pagare legato al numero di tracce dieci volte maggiore rispetto a prima sulla tabella e dall'altro di simulare lo scambio di beacon in modo più preciso attraverso una frequenza di 10 messaggi al secondo per ogni veicolo. Questa problematica è stata risolta grazie all'utilizzo del flag `--step-length 0.1` da specificare nella riga di comando all'avvio della simulazione con SUMO o da inserire nel tag <`time`> sul file con estensione *.sumocfg* che viene chiamato per avviare la simulazione. Naturalmente al crescere del numero di record, che con questa nuova configurazione va oltre i 25 milioni, cresce anche la dimensione dei file intermedi, come l'output XML di SUMO che arriva a circa 4 GB e quello CSV che supera i 2 GB, e del tempo richiesto dal calcolatore per elaborare e caricare sul database la grossa mole di informazioni presenti. In particolare, nella configurazione sulla quale sto lavorando (Apple MacBook PRO con 8 GB di memoria RAM e processore dual core Intel i5-5xxx) tra elaborazione e caricamento dei dati relativi alle sole tracce si va poco oltre i 20 minuti.

L'applicazione *SimulBeaconXchange* possiede una classe **Main** che rappresenta il corpo principale e altre due classi chiamate **Beacon** e **Trace** che rispettivamente modellano un beacon e una traccia. Il Main rappresenta il cuore dell'applicazione: comprende una serie di metodi utili per la connessione e la comunicazione col database e coordina le varie componenti mediante opportune chiamate per il completamento dell'intera operazione. Per finire questa breve panoramica sull'applicazione sviluppata che verrà discussa successivamente in modo più approfondito, oltre a quelle appena indicate è stata definita una quarta classe chiamata **Beaconing** che semplicemente modella un thread e ne definisce il "comportamento". L'applicazione, che a questo punto della discussione è nella sua fase di sviluppo iniziale, richiede che all'avvio le vengano forniti due parametri interi che rappresentano:

- il numero di thread da lanciare in parallelo;
- il range di trasmissione del segnale.

L'algoritmo di cui fa uso l'applicazione per la generazione dei beacon si basa sul concetto geometrico di distanza tra due punti e si serve dei valori delle coordinate spaziali x e y associate ad ogni veicolo. In particolare, per ogni traccia riferita ad un certo veicolo (sorgente) x all'istante t prelevata dalla base di dati viene valutato il vicinato di x attraverso il calcolo della distanza di quest'ultimo da tutti gli altri

veicoli presenti nello stesso istante della simulazione (destinatari) e nel caso in cui la distanza tra una coppia di nodi sia minore o uguale al range di trasmissione prescelto allora verrà simulata la generazione e la trasmissione di un beacon che parte dal veicolo sorgente e arriva al destinatario. Il pacchetto, definito attraverso una classe opportuna, è composto da una serie di campi e una volta creato viene direttamente caricato sulla tabella della base di dati chiamata *Beacons* che alla fine dell'operazione conterrà tutti i messaggi scambiati durante l'intero arco della simulazione da tutti i nodi.

La seconda parte di questo processo di studio si proietta sulla sola gestione del processo di beaconing. Mentre nella prima parte sono stati riscontrati dei tempi accettabili in questa i tempi si rilassano in maniera alquanto spropositata in quanto la procedura prima descritta deve essere attuata su ogni singola traccia e per tutti i veicoli tracciati in ogni dato `TIME_STEP`. Poiché nel caso di una simulazione completa il numero di tracce supera i 25 milioni, il tempo richiesto per l'elaborazione ed il caricamento non è indifferente. Volendo fare un esempio l'intera procedura avviata per un paio di ore comporta la generazione ed il caricamento di più di un milione di beacon per i soli primi 30 secondi della simulazione virtuale sullo scenario di Bologna che ha una durata totale (virtuale) maggiore di 60 minuti. Da qui è facile intuire che il tempo totale richiesto dall'intero processo ed il quantitativo di dati generati non è affatto trascurabile e per ovvi motivi prestazionali bisognerà quindi valutare eventuali ottimizzazioni nelle diverse fasi.

5.7 Valutazione delle prestazioni

Il processo che permette di ottenere i beacon sui quali andare poi a lavorare nella fase successiva per lo studio della protezione della rete VANET da eventuali malintenzionati necessita, come si è potuto vedere, di alcuni miglioramenti che permettano di ottenere delle prestazioni adeguate allo scopo.

La quantità dei dati generati ed elaborati nelle diverse fasi del processo non è indifferente ed è dell'ordine di diversi milioni tra tracce e beacon. Questo giustifica la ricerca di strategie che mirino al miglioramento di ognuna delle singole fasi che compongono il processo. In particolare per migliorare il tempo di esecuzione totale ho agito sia sullo script che sull'applicazione *SimulBeaconXchange*.

Nello script creato per il lancio del suddetto processo ho cercato di rendere più efficiente lo stadio iniziale di generazione delle tracce dei veicoli reali e fake andando a definire due funzioni ad hoc (una per le tracce reali ed una per le tracce fake) e utilizzando il suffisso "&" che consente di avviare i flussi di lavoro in parallelo. In questo specifico caso vengono avviati due flussi diversi: il primo si occupa della simulazione/generazione/salvataggio delle tracce reali mentre il secondo della simulazione/generazione/salvataggio delle tracce fasulle. Terminata questa fase sarà possibile prelevare ed elaborare le tracce già caricate sulla base di dati. Questo lavoro viene svolto dall'applicazione SBX. Poiché l'applicazione nel suo nucleo si preoccupa di simulare il processo di beaconing tra i veicoli presenti nella rete sulla base delle informazioni relative alle posizione dei veicoli stessi, essa deve effettuare una opportuna elaborazione delle tracce istante per istante per tutta la durata della

simulazione [t_o, t_f]. Questa operazione può essere espletata soltanto mediante la definizione di un ciclo all'interno del quale vengono effettuati i dovuti calcoli per ognuno dei timestep considerati e per tutti i veicoli presenti in quel dato timestep. Al fine del miglioramento di questa operazione o di questa raffica di operazioni, ho provveduto all'implementazione di una strategia multi-thread che si preoccupa di lanciare e gestire un pool di n thread alla volta (dove n è variabile e può essere scelto dall'utente attraverso la riga di comando al lancio dell'applicazione). In questo modo non si ha più un programma che elabora i dati timestep per timestep sequenzialmente ma si ha un programma che elabora n blocchi di timestep in parallelo. Il miglioramento ottenuto attraverso la messa in pratica di questa strategia è, come ci si può aspettare, non poco importante. Tuttavia risulta impossibile effettuare una stima a priori del tempo totale richiesto dall'applicazione SBX per simulare il beaconing tra tutte le autovetture che prendono parte alla simulazione in quanto i tempi di calcolo sono strettamente connessi al numero di tracce e alla loro distribuzione nei vari istanti di tempo ovvero alla densità dei veicoli. In termini generali è possibile affermare che esso dipende dalla dimensione del cluster (rappresentato dal blocco di tracce prelevate in ogni timestep) e dal numero di cluster da elaborare per l'intera durata della simulazione. Da un'analisi della complessità computazionale dell'algoritmo utilizzato sarà possibile desumere l'andamento di quella temporale. Assumendo che sia indicato con k il numero di timestep/cluster da valutare, che ogni cluster sia composto da n tracce (con n variabile in funzione della densità dei veicoli) e che saranno necessari $n * n - 1$ confronti (su ogni cluster) per verificare la vicinanza tra i nodi e generare (eventualmente) i beacon, allora è possibile dedurre che l'andamento del tempo sarà strettamente dipendente dalla dimensione n di ogni cluster considerato.

Lo studio e la messa in pratica di metodi che seppur nel breve periodo permettono di avere un risparmio di pochi secondi o nel caso migliore di qualche minuto, possono senza dubbio ricoprire un'importanza da non sottovalutare nel contesto dell'intera applicazione a causa della grossa mole di dati da elaborare. Le stesse considerazioni dovranno essere fatte quando più avanti saranno sviluppate le strategie per la protezione della rete e degli utenti della rete da possibili minacce derivanti dai cosiddetti attaccanti.

5.8 Attack model

Fino a questo momento è stata descritta, seppur in maniera piuttosto schematica, la prima fase dello studio condotto. Resta adesso da considerare e discutere l'ultima parte, quella rappresentata in rosso nella Figura 5.5, ovvero la simulazione degli attacchi veri e propri. In realtà la simulazione sarà parte del capitolo successivo mentre nel presente paragrafo viene trattato e sviluppato il cosiddetto attack model. Quest'ultimo in effetti è quel componente che sta a metà tra la generazione/memorizzazione dei beacon e la simulazione degli attacchi vera e propria. In generale questo modello descrive gli scenari di attacco che potrebbero essere messi in pratica dagli utenti maliziosi al fine di ledere la sicurezza della rete VANET e la privacy degli attori che ne fanno parte.

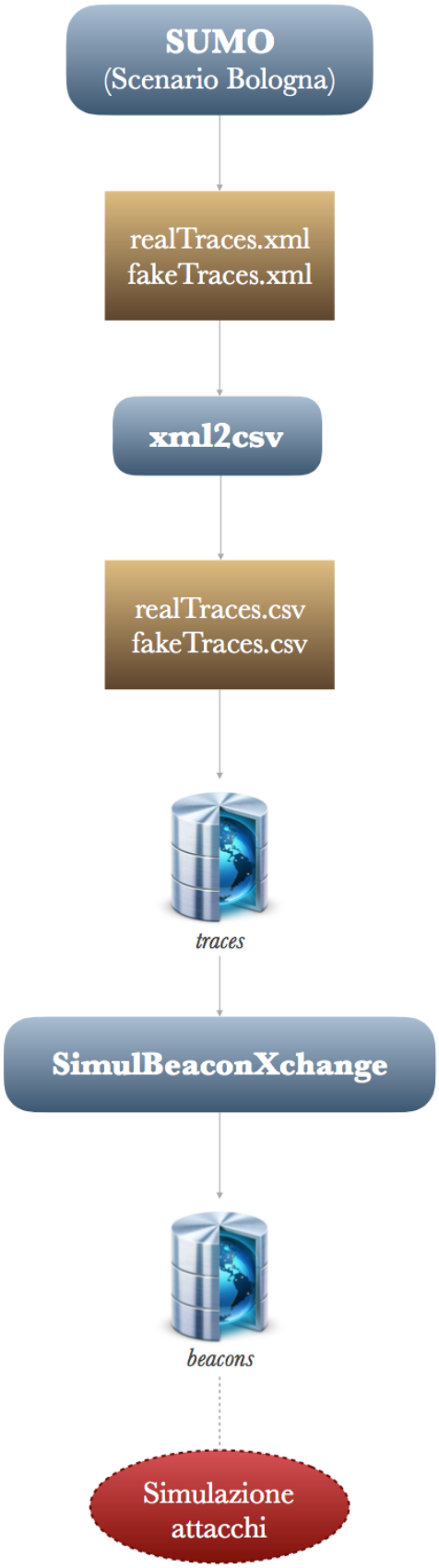


Figura 5.5: Rappresentazione grafica del processo

La creazione di un modello di attacco non è un'operazione semplice poiché è indispensabile calarsi nella parte e ragionare con la mente di un attaccante. La definizione di un possibile attacco deve essere preceduta dalla definizione dell'obiettivo che l'utente malizioso può voler raggiungere. Gli obiettivi possono essere diversi e le varie tipologie di attacco già considerate e descritte nel paragrafo 4.3 del capitolo precedente ne sono la dimostrazione. Proprio questa trattazione può essere scelta come punto di partenza per lo sviluppo di questo importante quanto indispensabile modello.

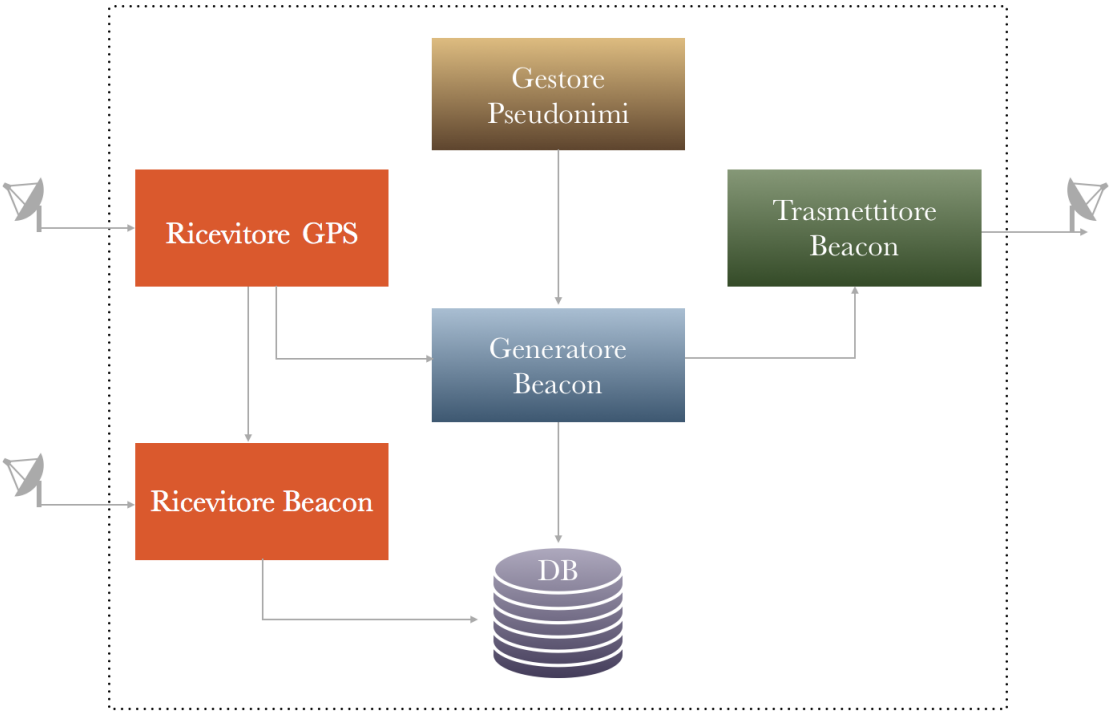


Figura 5.6: Modello della black box o OBU

5.8.1 Modello di partenza

Il modello presentato in questa sezione rappresenta schematicamente le componenti del sistema di comunicazione, la cosiddetta *black box* o *OBU*, utilizzata per rendere possibile lo scambio di messaggi tra le varie entità che compongono la rete VANET. La rappresentazione grafica risulta essere particolarmente interessante se non indispensabile in quanto consente di individuare uno ad uno i punti deboli di questo elemento chiave.

La componentistica della black box, mostrata in Figura 5.6, vede tra gli elementi più interessanti i ricevitori/trasmettitori di beacon e del segnale GPS ed il gestore degli pseudonimi per la protezione della *privacy* dell'utente. Ognuno di questi può rappresentare una vulnerabilità per il sistema.

- **Vulnerabilità #1: Ricevitore GPS**
  Il ricevitore GPS posizionato all'interno della OBU potrebbe "offrire" all'utente malintenzionato vari tipi di attacco che mirano sostanzialmente a limitarne le funzionalità. Tra questi troviamo:

  – *GPS Spoofing*
  Si tratta di una tipologia di attacco molto potente, già descritta nel capitolo precedente, attraverso il quale l'utente malintenzionato effettua in qualche modo la manipolazione del segnale GPS all'interno dell'area di attacco per far si che le informazioni catturate dai ricevitori GPS nelle OBU a bordo dei veicoli siano sotto il totale controllo dell'attaccante, il quale avrà come obiettivo principale quello di far credere che l'utente si trovi in una certa posizione ovviamente diversa da quella reale. Una possibile soluzione potrebbe essere quella di mantenere in una memoria le informazioni sulla posizione del veicolo istante per istante per far si che all'occorrenza, se un repentino cambio di segnale dovesse verificarsi, si possa procedere con una comparazione per valutarne la veridicità e la possibile correttezza. Un'ulteriore soluzione potrebbe essere quella di valutare istante per istante le informazioni sulla posizione che derivano dai veicoli del vicinato attraverso i beacon ricevuti e verificarne la compatibilità con quella del veicolo considerato. In entrambi i casi se la posizione attuale si discosta parecchio da quella che l'auto registrava nell'immediato passato o se risulta esserci un'incompatibilità sui dati relativi alla posizione del veicolo con quella del vicinato, allora vorrà dire che il nuovo segnale potrebbe non essere così genuino e il veicolo potrebbe quindi essere sotto attacco. Tuttavia la seconda soluzione considerata potrebbe poi non essere così efficace, basti pensare al caso in cui sia un'intera area ad essere colpita dall'attacco: tutti i veicoli coinvolti crederanno che l'informazione ricevuta sia corretta quando in realtà sappiamo non esserlo e solo quando questi usciranno dalla zona colpita dall'attacco potranno capire di aver ricevuto delle informazioni fino a quel momento corrotte.
  Il segnale GPS oltre alle informazioni sulla posizione trasmette l'informazione sul tempo generata mediante degli orologi atomici posti a bordo dei satelliti. Esso descrive con una certa precisione l'orario globale UTC e può essere usato per la definizione delle informazioni sul timestamp da inserire nell'apposito campo dei beacon. Inoltre esistono delle proposte in letteratura che prevedono l'utilizzo delle informazioni sull'orario di sistema come parametro principale per l'aggiornamento degli pseudonimi e la sua manipolazione potrebbe quindi portare a dei grossi rischi per la privacy degli utenti. In [68] vengono descritti alcuni tipi di attacco relativi al *GPS time spoofing* ed allo stesso tempo vengono presentate le possibili contromisure da adottare per rilevarli e/o prevenirli mentre in [69] viene presentata una possibile soluzione per prevenire e combattere un attacco replay mediante l'utilizzo delle informazioni trasportate dal segnale GPS captato dal ricevitore all'interno della On Board Unit.

  – *GPS Jamming*
  Un attaccante potrebbe disturbare o riprodurre con la stessa frequenza il

segnale GPS, che giunge sulla superficie terrestre dai satelliti in orbita con una intensità compresa tra i 10 e i 20 dB, semplicemente potenziandolo in modo tale che i ricevitori GPS dei nodi manipolati vengano disturbati ed effettuino la sincronizzazione con questo segnale potenziato e "corrotto" che farà naturalmente credere al veicolo di trovarsi in una posizione differente da quella reale. La possibile soluzione per cercare di intuire quando un nodo è soggetto a questo tipo di attacco è sostanzialmente simile a quella già descritta al punto 1 per il caso del GPS Spoofing.

– *GPS Blocking*
L'obiettivo principale di questo attacco è quello di disabilitare il ricevitore GPS attraverso la limitazione o il bloccaggio totale delle funzionalità dell'antenna utile per la ricezione delle informazioni sulla black box che quindi comporterà la totale assenza di informazioni sulla posizione del veicolo in input al generatore ed al ricevitore dei beacon. In questo caso la OBU, o meglio il ricevitore GPS, capterà subito l'assenza di segnale e dovrà quindi reagire in qualche modo mettendo in pratica un protocollo ad hoc che dica al sistema se terminare la generazione/trasmissione di beacon, se inviare dei messaggi di errore o ancora se inviare dei messaggi provando a stimare la propria posizione attraverso l'interpolazione dei dati relativi alla posizione dei veicoli del proprio vicinato prelevati dai beacon ricevuti durante il tragitto.

Ognuno di questi attacchi lede la funzionalità del ricevitore GPS e comporta un grave pericolo per gli utenti della rete VANET e non solo. Anche le compagnie assicurative negli ultimi anni hanno insistito molto attraverso campagne di sensibilizzazione e sconti sulla tariffa assicurativa, nell'intento di convincere i propri clienti ad installare la cosiddetta scatola nera all'interno dei veicoli per difendersi da possibili truffe ed allo stesso tempo offrire nuove soluzioni e servizi. Queste fanno fortemente uso del segnale GPS e come la OBU possono senza dubbio essere oggetto di attacchi. Numerosi studi e prove sono stati condotti nella speranza di sviluppare delle possibili soluzioni al fine di scongiurare attacchi al sistema GPS. [70] affronta ampiamente il problema del *GPS spoofing* attraverso la definizione di possibili modelli di attacco sia in ambito civile che militare e la valutazione dell'impatto che questo può avere sia nel caso di una sola vittima che di gruppi di vittime con l'obiettivo di ottenere una possibile contromisura per contrastare questa tipologia di attacco.

Altre possibili soluzioni in realtà esistono e a titolo di esempio si possono citare i cosiddetti *sistemi di navigazione inerziali*. Questi, seppur nella fase di avvio facciano un minimo uso del segnale GPS per effettuare una sorta calibrazione, basano il loro funzionamento sulla presenza di una piattaforma composta da processori, accelerometri e giroscopi che senza l'aiuto di riferimenti esterni si preoccupano di effettuare istante per istante i calcoli e di valutare la posizione e la velocità dei veicoli sui quali sono installati. La principale limitazione è purtroppo il costo. Questi sistemi, come molte soluzioni studiate e sviluppate in letteratura per la protezione della rete VANET, non possono essere prese in

considerazione a causa dei costi troppo elevati che purtroppo limitano fortemente la ricerca e lo sviluppo di nuove soluzioni atte ad offrire una maggiore sicurezza agli utenti di queste reti.

- **Vulnerabilità #2: Generatore di Beacon e Gestore degli Pseudonimi**
  Queste due unità sono di assoluta importanza per il contesto trattato in quanto hanno come obiettivo principale quello di assicurare la correttezza e la coerenza delle informazioni inserite sui beacon poi trasmesse agli altri veicoli e alle RSU.

  Se il gestore degli pseudonimi per qualche ragione non dovesse svolgere il suo lavoro correttamente, si potrebbe correre il rischio che le informazioni relative ai dati identificativi del veicolo non siano validi e coerenti. Per questo motivo un veicolo ogni volta che riceve un pacchetto da una terza parte dovrebbe dapprima verificare la validità dello pseudonimo e agire di conseguenza accettando o rifiutando il messaggio ricevuto. Questo schema apporterebbe però un sovraccarico non indifferente all'intero sistema poiché richiederebbe una valutazione one-by-one online e/o offline di ogni singolo messaggio e poiché la quantità di dati che ogni veicolo riceve non è indifferente, è facile intuire che le prestazioni ne potrebbero risentire pesantemente. Allo stesso tempo, se il generatore dei beacon dovesse subire un malfunzionamento per cause diverse, ci potrebbe essere il rischio che i dati inseriti nei campi quali velocità, posizione e timestamp siano errati e quindi anche qui come prima bisognerebbe valutare i dati contenuti sui messaggi ricevuti onde evitare dati appositamente manipolati trasmessi dai nodi malevoli o dai nodi sotto attacco.

  Gli attacchi di cui sono suscettibili queste due entità sono vari e sono per lo più connessi all'integrità ed alla coerenza dei beacon:

  - *Active Flooding*
    I beacon vengono creati e trasmessi da tutti i veicoli durante i loro spostamenti con una certa frequenza. Se per qualche ragione un attaccante riuscisse a modificare questo comportamento e riuscisse a forzare il modulo che si preoccupa di gestirne la generazione, allora ecco che potrebbe essere messo in atto un *flooding attack*. Basterebbe semplicemente "convincere" il modulo ad inviare con un'altissima frequenza i beacon sulla rete per causare un'inondazione di messaggi replicati e quindi uno spreco della banda a disposizione. Se poi questo tipo di attacco fosse replicato in larga scala su più nodi all'interno di una certa area della rete allora l'attaccante sarà riuscito certamente nell'intento di minimizzare le capacità già limitate di trasmissione della rete VANET generando un *Denial Of Service* che come noto è un attacco con un grande impatto difficile da contrastare. Purtroppo un attacco di questo genere potrebbe risultare disastroso poiché non bisogna dimenticare la ragione principale per la quale si vuol implementare, seppur con mille difficoltà, una rete di questo genere: se gli utenti non dovessero avere la possibilità di scambiarsi le informazioni di sicurezza allora verrebbe meno l'intento principale che, come detto più volte, è strettamente connesso al miglioramento della sicurezza stradale degli utenti.

– *Data Freshness*
Se nel punto precedente si è fatto riferimento alla frequenza di trasmissione dei beacon, in questo caso viene considerata la qualità e la frequenza di aggiornamento dei dati immessi sui beacon da trasmettere. Infatti il problema è relativo alla freschezza delle informazioni trasmesse ed alla frequenza di generazione dei beacon. Se la frequenza di trasmissione rimane inalterata e viene invece intaccata o ridotta al minimo la frequenza di generazione, gli scenari che ci si può aspettare di trovare sono due: nel primo caso il trasmettitore invia i messaggi solo quando questi sono aggiornati ed inseriti nella coda di invio mentre nel secondo caso il trasmettitore continua a trasmettere lo stesso messaggio secondo la frequenza di invio predefinita fino a che non arriva un nuovo beacon sulla coda. In pratica è facile provare che i due scenari sono molto simili con un'unica differenza: nel secondo caso i dati replicati inutilmente sulla rete andrebbero ad occupare parte della banda disponibile. In entrambi i casi infatti si ha che per un certo intervallo di tempo il beacon non viene mai aggiornato. Se il trasmettitore continua a replicare lo stesso messaggio fino al prossimo aggiornamento, i ricevitori degli altri veicoli facendo una verifica delle informazioni in esso contenute potranno facilmente accorgersi della non-freschezza dei dati: basterebbe valutare le informazioni relative al timestamp ed alla posizione per capire che quel beacon non è da considerarsi valido. Il problema potrebbe invece presentarsi nel momento in cui il nodo azzera la frequenza di trasmissione. In questo caso il dispositivo verrebbe bloccato con il risultato di non produrre e trasmettere più alcun beacon sul canale di comunicazione facendolo divenire una sorta di fantasma.

– *Collusion Attack*
Questo caso si riferisce ad un attacco per certi versi simile a quello visto al punto 1 ma con una forza distruttiva molto più importante. Se più veicoli colpiti da questo tipo di attacco dovessero inviare i beacon in maniera coordinata e con una frequenza costante agli altri veicoli ma con dei dati appositamente e volutamente errati, ecco che si otterrebbe un attacco di tipo *Sybil*. Allo stesso tempo si potrebbe pensare di manipolare un gruppo di nodi costringendoli a trasmettere, come nel punto 1, i messaggi con un'alta frequenza per ottenere stavolta un disastroso attacco *DDOS*.

Nel capitolo precedente sono state descritte le modalità di funzionamento di entrambi i tipi di attacco ed è stato messo in risalto come un gruppo di nodi che agiscono in maniera volutamente coordinata possano essere veramente nocivi per la rete VANET. Se con un attacco *Sybil* è possibile convincere i veicoli "vittima" di una certa situazione per far si che l'attaccante possa ad esempio liberarsi una zona della rete stradale per compiere un illecito, con un attacco *DDOS* si potrebbe buttar giù un'intera zona della rete che non permetterà nella maniera più assoluta ai nodi di poter comunicare. Entrambi gli attacchi essendo compiuti coordinatamente da un gruppo di daemon, avranno un impatto maggiore rispetto ad altri, saranno più difficili da rilevare e bloccare e permetteranno in molti casi agli

attaccanti di ottenere ciò che desiderano proprio a causa della difficoltà nel controllarli e combatterli.

- **Vulnerabilità #3: Trasmettitore dei beacon**
  Tra le componenti mostrate in Figura 5.6 bisogna adesso considerare il blocco verde che rappresenta l'importantissimo trasmettitore dei beacon. Questo, insieme ai due ricevitori, è senza dubbio uno degli elementi che se colpito da malfunzionamento o voluta manomissione può creare dei gravi problemi al comportamento generale del sistema di comunicazione. Esso, tra le altre cose, gestisce la potenza di trasmissione del segnale. Cosa potrebbe succedere se questo valore venisse compromesso? In caso di aumento i messaggi verrebbero inviati a tutti i nodi presenti anche al di fuori dell'area desiderata e prestabilita dallo standard, viceversa, nel caso di una diminuzione, verrebbe coperta un'area molto più ristretta rispetto a quella voluta dallo standard. Entrambe le situazioni sono critiche: se venisse aumentata spropositatamente la potenza di irradiazione del segnale, i beacon raggiungerebbero anche i veicoli non interessati a quelle informazioni perché distanti dalla zona da essi segnalata, mentre nel caso opposto i messaggi potrebbero non giungere a chi di competenza. Quest'ultima situazione potrebbe ad esempio causare degli incidenti o degli ingorghi poiché i veicoli destinatari dei messaggi potrebbero non essere avvertiti in tempo di una qualche situazione presentatasi sull'asse stradale in un certo istante di tempo. E se venisse azzerato o venisse manomessa l'antenna del trasmettitore? Queste due casistiche causerebbero un blocco nella trasmissione dei dati che, come già fatto notare in precedenza nel caso del Data Freshness, potrebbero trasformare in fantasmi i veicoli colpiti.

  Per concludere questa panoramica sui possibili problemi o vulnerabilità connessi a questo componente, consideriamo la possibilità di inserimento di un ritardo nella trasmissione dei messaggi. In questo caso i messaggi potrebbero essere memorizzati e volutamente invalidati per causare dei malfunzionamenti al sistema di comunicazione. Naturalmente i destinatari dei messaggi, verificando le informazioni ivi contenute potranno accorgersi dal timestamp della qualità dei dati e procedere quindi con la loro cancellazione.

  Fino a questo momento sono stati considerati soltanto le possibili criticità e le problematiche connesse al dispositivo di trasmissione. Tuttavia esso potrebbe all'occorrenza essere un elemento di aiuto per l'intero sistema. Talvolta potrebbe essere volutamente controllato e bloccato dal sistema stesso per evitare che certi messaggi vengano inviati agli altri nodi. Ad esempio, se per qualche ragione un messaggio dovesse restare sulla coda di invio per un lungo lasso di tempo tanto da divenire inutile per gli eventuali destinatari, una possibile strategia di soppressione potrebbe essere rappresentata da un servizio, che effettuando un controllo della coda di invio, ne valuta il contenuto e decide eventualmente se procedere o meno con il suo annullamento. In questo modo si evita che messaggi inutili vengano trasmessi occupando parte della banda a disposizione.

- **Vulnerabilità #4: Ricevitore dei beacon**
  Il ricevitore dei beacon è il blocco color arancio con l'antenna in Figura 5.6 che ha il compito di recepire le informazioni trasmesse da tutti gli altri veicoli presenti all'interno dell'area definita dal range di ricezione prestabilito. Questo componente, come gli altri considerati e valutati fin a questo momento, non è esente da problemi e possibili falle che potrebbero intaccare le funzionalità del sistema. Una vulnerabilità riguarda la possibile eliminazione parziale o totale dei messaggi ricevuti. Il dispositivo potrebbe ad esempio scegliere volutamente di scartare tutti quei messaggi che provengono da una certa sorgente o che comunque abbiano certe proprietà. Inoltre essendo questo modulo responsabile della ricezione dei beacon è normale che possa essere soggetto inconsapevolmente ad attacchi replay o spoofing.

  Dando uno sguardo alla Figura 5.6 si può notare come questo componente sia connesso con il blocco che rappresenta la memoria locale della OBU sulla quale vengono memorizzati i dati ricevuti e con il ricevitore GPS. Affinché i dati presenti sullo storage locale siano integri ed autentici è indispensabile che nella fase precedente a quella del salvataggio siano effettuati tutti i possibili controlli: dalla valutazione della validità del certificato usato per firmare il pacchetto, alla verifica dei dati inseriti nei campi del pacchetto. A tal proposito le informazioni ricevute dal ricevitore GPS sono estremamente preziose per la valutazione dei messaggi: se un attaccante dovesse trasmettere ad un veicolo dei messaggi con dati sulla sua posizione non congruenti con la posizione attuale allora si può procedere scartando il messaggio così da evitarne la memorizzazione. Inoltre è importante il valore nel campo relativo al timestamp. Il modulo di ricezione per una maggiore sicurezza potrebbe all'occorrenza contenere un circuito hardware ad hoc o semplicemente un clock personale che gli consenta di valutare la freschezza dei dati ricevuti. A causa dell'intasamento della rete, a seguito ad esempio di un possibile attacco DOS in corso, potrebbe accadere che i messaggi arrivino con dei ritardi consistenti alle loro destinazioni. Il valore del timestamp inserito dalla sorgente e valutato con un check dal destinatario, consentirà ai ricevitori dei beacon di riconoscere quali informazioni debbano essere valutate ed eventualmente memorizzate e quali no.

  Naturalmente bisognerà definire gli step necessari da seguire per le valutazioni dei messaggi: in primis deve essere valutata l'integrità e l'autenticità del beacon, poi si potrebbe valutare la freschezza dei dati tramite il timestamp ed infine i dati sulla posizione geografica della sorgente. Idealmente questi appena indicati potrebbero essere senza dubbio degli step importanti per schivare i messaggi "inutili" anche se non proteggono totalmente il sistema da possibili altri attacchi ai quali è esposto. Ad esempio, un attaccante potrebbe sfruttare delle risorse esterne, come un PC, o interne, come le RSU o le altre OBU, capaci di comunicare sulle stesse frequenze dei canali indicati dallo standard WAVE per far si che un nodo vittima venga inondato di informazioni inutili con l'intento di limitarne le capacità e le risorse così da renderlo inattivo.

- **Vulnerabilità #5: Memoria**
  Ogni OBU deve contenere al suo interno una memoria di modeste dimensioni

per il salvataggio delle informazioni ricevute e già precedentemente validate. I dati presenti al suo interno sono molto importanti in quanto possono risultare utili per un processamento a posteriori al fine di rilevare delle criticità o semplicemente per la valutazione di situazioni particolari come un incidente. In questo modo le compagnie assicurative potrebbero avere un maggior controllo sui dati e sui propri clienti ed evitare possibili truffe a loro carico.

Purtroppo anche la memoria potrebbe essere soggetta ad attacchi. Un attaccante potrebbe voler non solo effettuare un'invalidazione dell'intero contenuto o di parte di esso ma anche aggiungere dei beacon aventi delle proprietà studiate appositamente per compiere un illecito infrangendone l'integrità. Per evitare che chiunque possa accedervi liberamente, una possibile soluzione potrebbe essere quella della cifratura dei dati, anche se da sola non assicura la totale protezione da eventuali attacchi. Infatti se qualsiasi applicazione per ogni operazione di lettura/scrittura di un dato in memoria mediante delle specifiche funzioni dovesse avere libero accesso alle chiavi segrete allora il livello di sicurezza ottenuto non sarà affatto soddisfacente in quanto i segreti, e di conseguenza i dati, sarebbero eccessivamente esposti agli occhi di possibili malintenzionati.

La *sicurezza delle informazioni* salvate in una memoria locale così come in una base di dati accessibile tramite applicazioni esposte sul web è un concetto fondamentale. Per questo motivo ogni qual volta si ritenga necessaria la protezione dei dati è giusto ricorrere all'implementazione di metodi opportuni o di schemi ad hoc che aiutino a limitare i danni nel caso in cui un attaccante riuscisse ad accedervi. Tuttavia esistono delle *linee guida* o *procedure* che se seguite e messe in atto unitamente ad alcuni controlli di sicurezza, possono aiutare gli utenti a proteggere i propri dati. Tra questi uno dei più importanti è senza dubbio il *controllo di accesso* il quale permette di verificare l'identità e i privilegi degli utenti per assicurarsi che questi non abbiano dei comportamenti illeciti e che stiano accedendo esclusivamente alle proprie informazioni. Il processo di autenticazione che consentirebbe l'accesso alla memoria alle compagnie assicurative o alle Forze dell'Ordine in certe situazioni attraverso le proprie credenziali già caricate e salvate in modo sicuro all'interno del dispositivo, potrebbe basarsi ad esempio su un protocollo di handshake per rinforzare il sistema e, allo stesso tempo, la OBU potrebbe usare le proprie chiavi opportunamente "caricate" sul modulo di sicurezza presente a bordo del veicolo al fine di firmare i dati da esportare per garantirne l'autenticità. Inoltre l'estrapolazione dei dati dalla OBU, che potrebbe essere effettuata sia via wireless che wired, deve essere resa sicura mediante l'utilizzo di protocolli opportuni per evitare intercettazioni da parte di utenti non fidati che potrebbero anche alterarli mettendo in atto un attacco man-in-the-middle. Per finire, la presenza di particolari dispositivi software o hardware che effettuano un monitoraggio del sistema e permettono di registrare eventuali attacchi o intrusioni e l'assenza di backdoor per un accesso univoco ai dati completano lo schema e donano un livello di sicurezza maggiore al sistema di storage della On Board Unit.

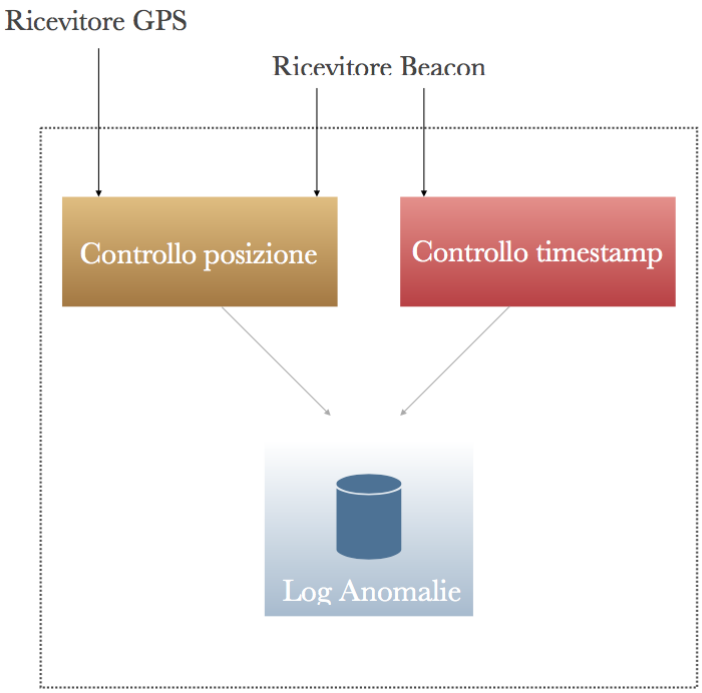


Figura 5.7: Modello dell'*Anomaly Checker*

5.8.2 Estensione del modello

Viene adesso valutata una possibile estensione del modello di base, già ampiamente considerato e rappresentato in Figura 5.6, che va ad aggiungere un ulteriore modulo chiamato *Anomaly Checker*. Questo blocco, rappresentato graficamente in Figura 5.7, può essere visto come un controllore al quale fanno riferimento le OBU al fine di considerare e valutare eventuali anomalie presentatesi durante il processo di Inter Vehicle Communication. Come si può notare dalla figura, questa scatola di controllo riceve idealmente dalla OBU due possibili valori in ingresso: la posizione attuale del veicolo rilevata dal GPS ed il beacon vero e proprio ricevuto dal blocco di ricezione. Queste informazioni sono particolarmente utili per la rilevazione di possibili anomalie o errori che verrebbero eventualmente riportate sulla memoria del dispositivo stesso. Il messaggio, una volta validato e verificato, viene spedito per ulteriori analisi all'Anomaly Checker che internamente è costituito da tre elementi:

- *Controllore della posizione*
  Effettua una verifica sulla qualità delle informazioni contenute nei campi del beacon ed in particolare valuta i dati relativi alla posizione del veicolo che lo ha generato per assicurarsi che esso risieda all'interno dell'area prestabilita dallo standard. Se il veicolo riceve un beacon da un altro veicolo che afferma di essere da tutt'altra parte rispetto a quella "accettabile" allora viene rilevata un'anomalia che verrà salvata sulla memoria locale e causerà lo scarto dell'informazione ricevuta;

- *Controllore del timestamp*
  Effettua il controllo della qualità dei dati ricevuti tramite i beacon dagli altri

veicoli. Questo riceve interamente il beacon e, al fine di valutare la freschezza delle informazioni tramite la verifica del timestamp presente nell'apposito campo, avrà un clock interno affidabile che gli permetterà di effettuare una comparazione sicura: se i dati ricevuti risultano essere troppo datati allora si procede con la segnalazione e la memorizzazione dell'anomalia;

- *Log delle anomalie*
  Questo componente non è altro che una memoria di dimensione opportuna installata nell'Anomaly Checker che sarà utilizzata come contenitore per il salvataggio di tutte le anomalie rilevate durante le due fasi di controllo. Naturalmente è sufficiente che almeno uno dei due controlli dia esito negativo per generare un'anomalia.

Il cosiddetto *Anomaly Checker* rappresenta senza dubbio un dispositivo molto importante che, come già detto più volte, consente di effettuare delle valutazioni dei messaggi in ingresso alla OBU. Il processo di verifica deve essere reso più efficiente possibile per evitare accodamenti e perdite di dati poiché non bisogna dimenticare la grossa mole di informazioni che un veicolo può ricevere, in relazione allo stato della circolazione veicolare e quindi del numero di veicoli che risiedono istante per istante nell'area ricoperta dai range di trasmissione/ricezione. Inoltre si è dato per certo che la validazione del messaggio mediante la verifica del certificato sia stata effettuata ancor prima che i dati giungano all'Anomaly Checker. Se così non fosse al modello qui presentato si potrebbe aggiungere un nuovo blocco che, unitamente ai due controllori, verifichi l'autenticità dei messaggi e segnali quindi le eventuali anomalie riscontrate. La verifica dell'autenticità e dell'integrità dei dati comporta la valutazione dello pseudonimo inserito all'interno del beacon da colui che lo ha generato. Poiché ad oggi non esiste una *Certification Authority* con un'infrastruttura ben definita che rappresenti il punto di riferimento al quale afferire per la validazione di questi dati, è difficile parlare di prestazioni in quanto l'overhead apportato da questo ulteriore controllo non è indifferente. Idealmente è facile intuire che anche questo processo debba essere ottimizzato al massimo onde evitare un decadimento prestazionale dell'intera architettura.

La Figura 5.7 tuttavia non mostra alcun dispositivo di trasmissione dei dati e quindi ciò lascia presagire che tutte le informazioni vengano solamente salvate in locale sul dispositivo e possibilmente valutate in casi particolari. A tal proposito, una possibile miglioria potrebbe essere quella di inserire un trasmettitore ad esempio che effettui l'invio di segnalazioni alle RSU nel momento in cui avviene il rilevamento di una certa anomalia in modo da informare gli altri veicoli della possibile situazione di pericolo individuata.

5.9 Logging dei messaggi

Nei primi paragrafi del presente capitolo è stata presentata la fase iniziale del processo di creazione delle tracce a partire dal simulatore di traffico e di creazione dei beacon grazie all'applicazione *SimulBeaconXchange* appositamente sviluppata. Questi

beacon, salvati momentaneamente su una tabella della base di dati, rappresentano le informazioni che è possibile trovare sulle memorie interne alle OBU.

Il *logging* è un'operazione che consiste nella selezione e nel successivo salvataggio dei messaggi sulla memoria delle *black box*. In sostanza esso simula il processo di salvataggio delle informazioni sulle EDR dei veicoli. La selezione dei messaggi, che può essere ad esempio eseguita da un algoritmo di scarto ad hoc che ne verifica le proprietà e ne decide il destino, è molto importante per mettere in atto uno scenario più realistico possibile. Questo step dovrebbe seguire il processo di generazione dei beacon e ricevere come dati di input per ogni veicolo, tutti i messaggi ricevuti istante per istante.

Alla fine del processo di logging si avranno idealmente due liste di oggetti che rappresentano rispettivamente i messaggi loggati (validi) e i messaggi scartati (non validi). Come visto nel Paragrafo 5.6, l'area di ricezione/trasmissione di un messaggio può essere approssimata all'area di un cerchio avente raggio uguale al range di propagazione del segnale e come centro il veicolo stesso.

Notoriamente le reti possono generare delle perdite di informazioni che possono affliggere i pacchetti durante il loro transito sul canale di comunicazione a causa di diversi possibili ritardi e/o dello stato dei link. Inoltre non bisogna dimenticare il contesto applicativo delle reti VANET: a causa dei disturbi che possono preoccupare i canali WAVE e degli ostacoli fisici, come gli edifici, che possono disturbare la propagazione del segnale nell'aria, si ha un'alta probabilità di errori nelle comunicazioni che possono quindi causare malfunzionamenti e generare errori nella comunicazione interveicolare. Nello scenario considerato finora è stato modellato un sistema perfetto in cui tutti i messaggi arrivano sempre a destinazione e tutti i messaggi sono loggati con successo sulle OBU. Purtroppo questo è il caso ottimo e al fine di simulare uno scenario realistico (ideale) è necessaria la definizione di una strategia che individui possibili messaggi che non saranno mai loggati dai destinatari e che saranno quindi virtualmente scartati. La strategia di scarto può basarsi ad esempio su un algoritmo *Distance Based* ovvero sulla distanza tra il nodo sorgente ed il nodo destinatario: maggiore è la distanza che intercorre tra i due veicoli maggiore sarà la probabilità che il messaggio non giunga a destinazione o che non arrivi "integro" al destinatario e quindi maggiore è la probabilità che non venga mai loggato dal nodo destinatario.

5.9.1 Implementazione della strategia di logging dei beacon

L'applicazione SBX sviluppata si limita a simulare lo scambio di beacon nel caso ottimo in cui non ci siano errori e a salvare su una tabella della base di dati tutti i beacon scambiati tra tutti i nodi in tutti gli istanti tempo per l'intera durata della simulazione. Bisogna adesso aggiungere una strategia di scarto che, in maniera casuale, individui eventuali pacchetti non salvati dai veicoli destinatari. L'idea proposta nel precedente paragrafo si basa sostanzialmente sul concetto di distanza tra due nodi che intendono scambiarsi delle informazioni: il range di trasmissione, impostato ad un certo valore, definisce un'area, che verrà suddivisa in 10 sezioni circolari di uguale spessore, avente come centro il veicolo che deve ricevere il messaggio. Il valore standard del range di trasmissione per l'applicazione, come detto più volte,

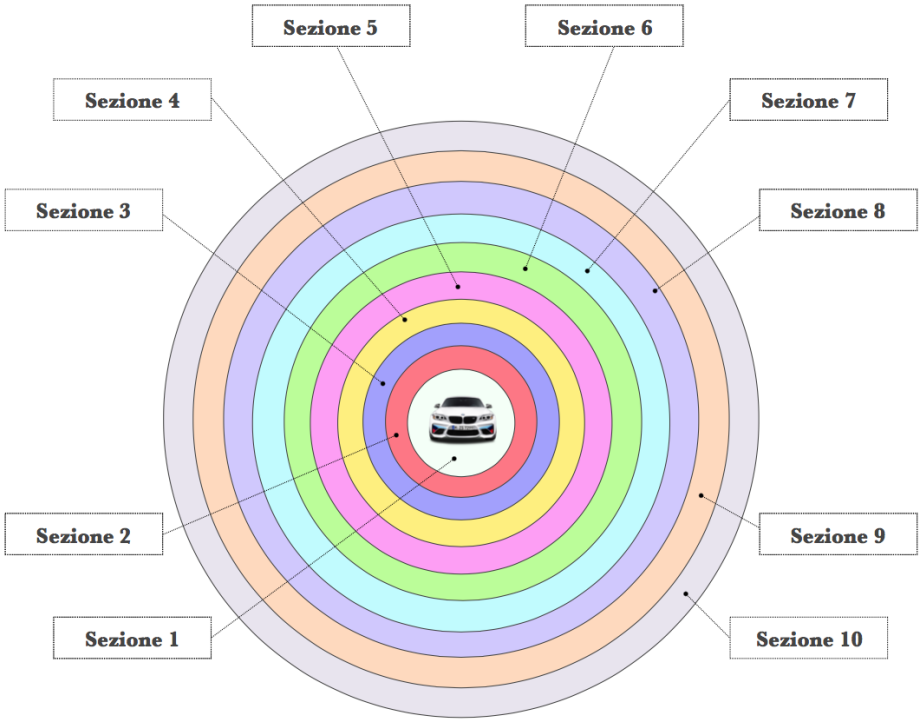


Figura 5.8: Range di ricezione di un veicolo con relative sezioni

è pari a 300 metri. Ciò vuol dire che per ogni veicolo in ogni istante di tempo verrà generata un'area stratificata composta da 10 corone circolari aventi una sezione di 30 metri ciascuna ($^{300}/_{10}$) (Figura 5.8). Più si andrà verso la corona esterna e minore sarà la probabilità che un certo messaggio possa essere loggato dal destinatario.

Una semplice strategia applicativa consiste nell'associare ad ogni sezione una percentuale che rappresenta la probabilità di recapito del messaggio. In particolare, andando dalla parte più interna a quella più esterna si avrà una probabilità di logging dei pacchetti che andrà via via scemando. Quindi se una fascia avesse il 30% di probabilità associata allora vorrebbe dire che ogni pacchetto che rientra idealmente in quell'area avrà il 30% di probabilità di essere salvato correttamente dal destinatario sulla propria memoria.

Nel caso specifico dell'applicazione SBX la configurazione scelta è la seguente:

- *Sezione #1:* area più interna (fino a 30 metri di distanza) $\Longrightarrow$ 100%
- *Sezione #2:* da 30 a 60 metri $\Longrightarrow$ 90%
- *Sezione #3:* da 60 a 90 metri $\Longrightarrow$ 80%
- *Sezione #4:* da 90 a 120 metri $\Longrightarrow$ 70%
- *Sezione #5:* da 120 a 150 metri $\Longrightarrow$ 60%
- *Sezione #6:* da 150 a 180 metri $\Longrightarrow$ 50%
- *Sezione #7:* da 180 a 210 metri $\Longrightarrow$ 40%

- *Sezione #8:* da 210 a 240 metri $\Longrightarrow$ 30%
- *Sezione #9:* da 240 a 270 metri $\Longrightarrow$ 20%
- *Sezione #10:* da 270 a 300 metri $\Longrightarrow$ 10%

Affinché la strategia proposta possa essere attuata, l'applicazione dovrà essere ulteriormente estesa con un blocco di codice che effettui le dovute valutazioni e gestisca opportunamente i beacon per ogni istante di tempo e per ogni veicolo presente nella simulazione. In particolare essa effettuerà i seguenti passaggi:

- valutazione del beacon generato e recapitato all'istante di tempo t ad un certo veicolo v;
- definizione logica del cerchio stratificato, come quello rappresentato in Figura 5.8, e valutazione della sezione nella quale "ricade" il beacon considerato;
- calcolo della probabilità di logging del messaggio e conseguente salvataggio/eliminazione in base alla percentuale associata.

Per simulare il salvataggio e lo scarto dei vari messaggi, l'idea è quella di agire direttamente sul blocco di codice che gestisce la generazione dei beacon all'interno della classe Beaconing e procedere con l'eventuale salvataggio sulla tabella *Beacons* dei soli messaggi che soddisfano i criteri prima indicati. Alla fine dell'intera operazione ci ritroveremo quindi ad avere una tabella Beacons contenente soltanto la totalità dei messaggi che hanno raggiunto correttamente la propria destinazione e che quindi sono virtualmente salvati sulle memorie interne alle OBU dei veicoli destinatari.

Capitolo 6

Implementazione e simulazioni

Lo studio condotto fino a questo punto della trattazione prende in considerazione in maniera completa ogni singolo aspetto di una Vehicular Ad-hoc NETwork: dalle esigenze costruttive alle caratteristiche, dai pregi ai difetti, dalle componenti fisiche e strutturali ai protocolli sviluppati, dalla comunicazione interveicolare all'infrastruttura di sicurezza. Come si è potuto notare, l'implementazione di una rete VANET non è un'operazione di poco conto: le limitazioni dovute alla difficoltà di sviluppo in un contesto reale, ai costi dei componenti, alle esigenze di calcolo e a quelle strutturali sono degli ostacoli che purtroppo ne hanno rallentato pesantemente l'iter della crescita.

Il capitolo precedente è stato dedicato ampiamente all'architettura di alto livello dell'applicazione sviluppata per simulare le funzionalità principali di una VANET. Sono state inoltre descritte schematicamente le varie componenti utili per l'espletamento delle funzioni proprie di una rete di comunicazione ed allo stesso tempo sono state evidenziate grazie all'attack model le criticità e le possibili vulnerabilità di cui può soffrire.

Nel presente capitolo la trattazione continuerà scendendo più nello specifico e considerando in maniera completa e dettagliata ogni singolo componente dell'architettura implementata. Inoltre verranno nuovamente presi in considerazione i vari tipi di attacco e traendone spunto si cercherà di simulare degli scenari ad hoc al fine di verificare le conseguenze e valutare le possibili contromisure da mettere in atto per la protezione della rete e dei suoi utenti dai cosiddetti attaccanti. Infine verrà discusso il problema della sicurezza del processo di beaconing con riferimento agli aggiornamenti più recenti dello standard IEEE 1609.2 e saranno valutati i risultati ottenuti e l'efficacia delle contromisure sviluppate per la mitigazione degli attacchi.

6.1 La sicurezza nel beaconing

Il beaconing è un processo di fondamentale importanza per la rete VANET poiché consente ai nodi di scambiarsi informazioni relative alla viabilità stradale e di segnalare eventuali situazioni di pericolo. Queste informazioni consentono ai conducenti di avere una maggiore consapevolezza delle condizioni del traffico e di adottare le

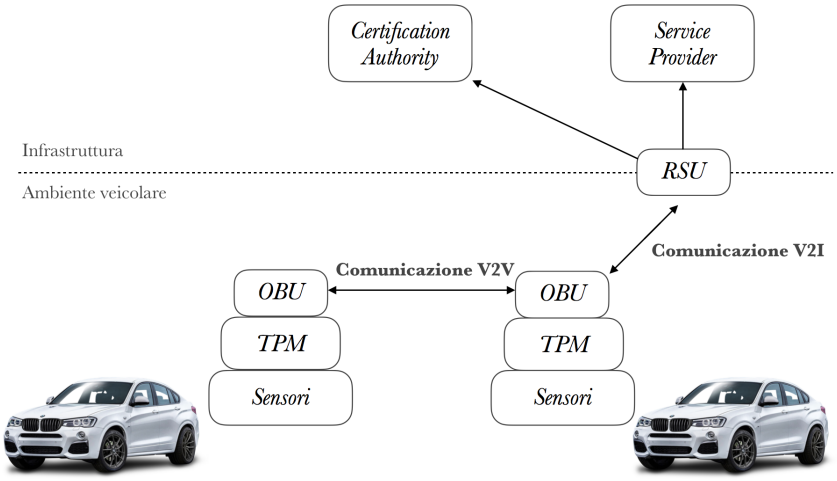


Figura 6.1: Infrastruttura rete VANET e modello di comunicazione V2V-V2I

necessarie azioni nei momenti opportuni. I beacon vengono trasmessi con una certa frequenza da tutti i componenti della rete e rappresentano l'elemento base per molte applicazioni. Questo processo però deve essere reso sicuro per evitare situazioni che possano infrangere la sicurezza degli utenti. Le principali strategie di protezione si basano sull'uso di certificati per assicurare l'autenticità e l'integrità dei pacchetti. Tuttavia il dispendio di risorse legato al processo di verifica e validazione dei dati ricevuti non è indifferente specie nel momento in cui un veicolo si trova in una zona ad alta densità di traffico. Basti pensare che la OnBoard Unit dovrebbe essere equipaggiata con un processore da 400MHz che richiederebbe all'incirca 20 millisecondi per verificare una sola firma. A questo punto piuttosto che considerare il concetto di sicurezza come un caso a sé bisognerebbe progettare delle strategie che offrano un certo grado di protezione unitamente a delle prestazioni adeguate al contesto applicativo.

Se il beaconing non fosse protetto a dovere gli attaccanti, sia interni che esterni, potrebbero violare la sicurezza degli utenti e della rete mediante i vari tipi di attacco già esposti nel Capitolo 4. Lo standard IEEE 1609.2 propone un meccanismo di base per rendere sicura la rete VANET che fa uso della crittografia a chiave pubblica con ECC ma non prevede ancora alcuna strategia per la protezione della privacy degli utenti. Esso definisce il formato dei messaggi, contiene i metodi per rendere sicura la gestione dei messaggi e descrive i requisiti necessari per supportare le funzioni di sicurezza. Le proprietà di questo meccanismo sono riassunte nei seguenti punti:

- I veicoli sono in possesso di una coppia di chiavi e di un certificato rilasciato da una Certification Authority.
- Ogni nodo firma i propri messaggi con la propria chiave privata mentre i ricevitori utilizzeranno la corrispondente chiave pubblica per procedere con la verifica. Il certificato contenente queste informazioni è allegato al messaggio stesso.
- I beacon che non conterranno un certificato valido o che non risulteranno integri saranno scartati dai ricevitori.

- Ogni beacon contiene un timestamp o un numero di sequenza per prevenire attacchi di tipo replay.
- I dati relativi alla sicurezza associati ad ogni veicolo sono salvati nel TPM a bordo del veicolo stesso al fine di evitare manomissioni (Figura 6.1).

Purtroppo l'inserimento del certificato sul beacon e la verifica successiva dei messaggi causano rispettivamente un incremento delle dimensioni dei pacchetti e una crescita dell'overhead della rete (maggiore è la dimensione dei pacchetti maggiore sarà la banda occupata e il carico del canale DSRC dedicato a parità di numero di pacchetti). Inoltre i nodi richiederanno una certa potenza di calcolo aggiuntiva per poter valutare efficacemente e senza particolari sovraccarichi la mole di dati ricevuti. Per rendere più leggero e veloce il processo di protezione della rete VANET appena proposto si potrebbe ad esempio pensare di agire sul certificato omettendolo dal contenuto del messaggio o evitandone la verifica una volta che il messaggio giunge a destinazione (in casi particolari). Naturalmente l'eliminazione di uno o più elementi deve essere valutata e verificata attentamente ed in [71] vengono esposte le diverse possibilità per delineare una strategia che possa unire la sicurezza della rete e degli utenti al rendimento dell'intero sistema.

Nel marzo del 2016 lo standard IEEE 1609.2 è stato rivisto ed aggiornato e tra le novità è possibile trovare adesso una definizione del formato del certificato rilasciato dalla CA e del formato delle CRL (*Certificate Revocation List*). Inoltre è stato introdotto il protocollo SPDU (*Secure Protocol Data Unit*) che protegge il payload dei messaggi ed un altro protocollo che gestisce la distribuzione P2P (peer-to-peer) dei certificati per consentire ai vari nodi di imparare i certificati sconosciuti. Un Signed PDU dovrebbe contenere le seguenti informazioni:

- almeno uno tra payload e hash del payload;
- identificativo del Service Provider per indicare i permessi;
- header addizionali che contengono le informazioni sulla data di creazione e sulla validità;
- firma;
- riferimento al certificato.

In tutto ciò non è ancora stata considerata un'altra nota criticità: la privacy degli utenti. Nel Capitolo 2 è stata descritta ampiamente l'unica strategia valida avanzata nel corso degli ultimi anni per la protezione della privacy, ovvero quella che fa uso degli *pseudonimi*. Diversi schemi per l'ottimizzazione di questo processo sono stati sviluppati poiché esso va ad influire ulteriormente sul carico della rete e quindi sulle prestazioni generali della rete. Uno pseudonimo non è altro che una chiave temporanea associata al veicolo e certificata da una terza parte fidata che cambiando ad intervalli regolari funge da maschera permettendo di celare a terzi l'identità reale dell'utente o del veicolo. In [72] vengono proposte diverse possibili migliorie per rendere più efficiente la gestione degli pseudonimi e vengono anche

analizzati e valutati mediante delle simulazioni l'impatto e i limiti delle tecniche presenti in letteratura sulla sicurezza dei trasporti e sul miglioramento della privacy degli utenti. In [73] viene invece presentato un nuovo protocollo che fa uso di ben due pseudonimi per ogni veicolo, uno con vita breve (temporaneo) ed uno fisso; da qui il nome *protezione della privacy a due livelli*.

Fintantoché una Vehicular Ad hoc Network non rispetterà tutti i requisiti di sicurezza indispensabili per proteggere gli utenti ed il processo di beaconing, non sarà possibile procedere con l'implementazione e la messa in pratica in larga scala. Tutte le componenti in gioco devono essere valutate attentamente per rilevare eventuali vulnerabilità "nascoste" da proteggere mediante lo sviluppo di opportune contromisure onde evitare che utenti malintenzionati, interni o esterni, possano sfruttarle creando dei disagi non indifferenti. Poiché gran parte delle criticità derivano da utenti esterni, un meccanismo di autenticazione e registrazione potrebbe essere sufficiente per contrastare molti dei possibili attacchi. L'utilizzo di firme digitali per autenticare gli utenti nella rete VANET permetterebbe quindi, nel caso ideale di un attacco, di identificare il nodo malizioso. La vera sfida futura sarà però quella di riuscire a trovare un punto comune tra tutte le organizzazioni coinvolte nello sviluppo della VANET al fine di definire uno standard valido, unico, comune a tutti i paesi, che indichi le direttive funzionali agli utilizzatori finali ed allo stesso tempo li protegga da eventuali *attacchi*.

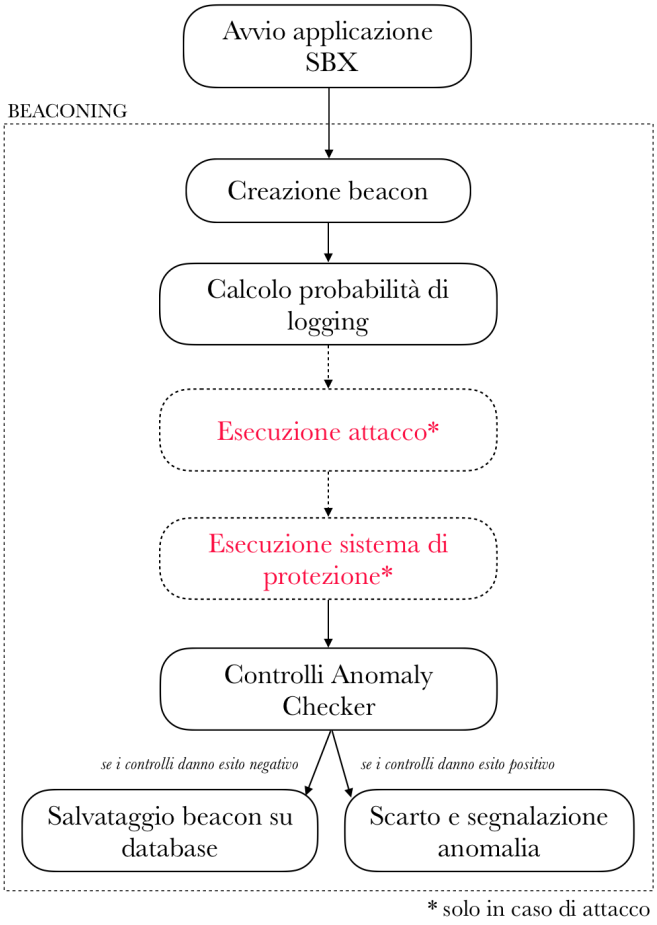


Figura 6.2: Workflow applicazione *SimulBeaconXchange*

6.2 Estensione applicazione SBX

L'applicazione SBX descritta fino a questo momento implementa, seppur idealmente, la funzionalità fondamentale di una rete VANET ovvero la continua trasmissione di informazioni e messaggi attraverso l'ormai noto processo di beaconing. A questo punto è indispensabile procedere con un ulteriore "ampliamento" delle funzioni sviluppate al fine di poter mettere a disposizione un ambiente virtuale attraverso il quale sia possibile simulare gli attacchi a cui possono essere soggetti le diverse componenti della On Board Unit e di conseguenza gli utenti della rete. La definizione di questo ambiente è resa possibile grazie all'utilizzo di alcune nuove funzioni definite all'interno di una nuova classe chiamata *Attacks* le quali implementano sia gli attacchi veri e propri che le strategie utili per la loro mitigazione. In totale potranno essere simulati ben 13 diversi scenari di attacco individuati mediante l'utilizzo di un codice univoco che una volta passato come parametro all'applicazione SBX ne consentirà l'identificazione e l'esecuzione. Ognuna delle tredici simulazioni sarà ampiamente trattata nel paragrafo successivo e restituirà alla fine della sua esecuzione un resoconto dell'operazione all'interno di un apposito file di log grazie al quale sarà possibile conoscere la percentuale dei pacchetti rilevati dal sistema di protezione relativo all'attacco scelto e quindi l'esito del sistema di difesa adottato.

Fino ad ora la classe Beaconing ha avuto il solo compito di simulare le proprietà del processo omonimo. Adesso con la presenza delle simulazioni degli attacchi il suo comportamento dovrà essere rivisto e modificato in quanto essa non si limiterà più alla sola creazione e trasmissione dei beacon ma dovrà richiamare al momento opportuno sia l'attacco "richiesto" che la relativa strategia sviluppata per una eventuale rilevazione al fine di poter consentire ai nodi vittima di salvare sulla propria memoria solo ed esclusivamente i messaggi "buoni" evitando e scartando quelli che invece potrebbero contenere delle informazioni manipolate. Tuttavia, come sarà possibile notare dai risultati dei test condotti per ognuno degli attacchi, non tutti i dati manipolati dagli attaccanti potranno essere correttamente individuati e scartati dai sistemi di rilevazione sviluppati che quindi non potranno offrire una protezione totale agli utenti della rete. La difficoltà della rilevazione di certe azioni di attacco nel contesto virtuale è strettamente correlata al fatto di non poter fare affidamento ad elementi o strumenti che siano diversi dai soli controlli sui dati contenuti all'interno dei messaggi scambiati.

In riferimento a quanto detto finora in merito ai sistemi di protezione sviluppati è bene precisare che la gran parte di questi sono stati pensati nell'ottica di una *rilevazione a caldo* di un possibile attacco. Ogni nodo localmente dovrà eseguire i dovuti controlli per valutare la coerenza e la validità dei dati ricevuti attraverso i beacon dagli altri utenti della rete. In particolare, la strategia di protezione implementata per ognuno degli attacchi verrà lanciata prima che venga eseguito il caricamento dei beacon sulla base di dati grazie ad una funzione dedicata. Così facendo solo i beacon che avranno superato i controlli saranno passati a questa funzione per il loro salvataggio in memoria mentre quelli che sono stati etichettati come "manipolati" verranno opportunamente scartati e segnalati. Bisogna però precisare che non tutte le tipologie di attacco potranno fare affidamento ad un "semplice" controllo locale e, per forza di cose, alcune di queste richiederanno l'accesso alla tabella del database che memorizza tutti i beacon scambiati da tutti i nodi per l'intera durata della

simulazione affinché possano condurre un'analisi *globale* per provare ad individuare la presenza di attaccanti sulla rete. La sostanziale differenza tra i due modelli di protezione può essere riassunta mediante la definizione di due diverse categorie alle quali è possibile associare i vari schemi di difesa sviluppati. La prima categoria è rappresentata dai sistemi di protezione locali mentre la seconda da quelli globali. Le due classi si differenziano per la modalità di azione adottata da ogni schema: i sistemi di protezione locali possono essere messi in atto in modo automatico ed autonomo da ogni nodo e si limitano ad eseguire delle verifiche di coerenza sui dati presenti all'interno dei campi dei beacon ricevuti mentre quelli globali si affidano a delle strategie che richiedono l'accesso alla totalità dei beacon generati per eseguirne l'interpolazione dei dati presenti al loro interno al fine di consentire il rilevamento di un particolare attacco. Da quanto detto potrebbe sorgere un dubbio sull'effettiva rapidità e quindi validità di un sistema di rilevazione globale. Infatti esso non consentirebbe ai nodi di proteggersi *direttamente* dagli attaccanti ma dovrebbe essere un'entità esterna ad agire "dietro le quinte" e ad individuare ed estromettere dalla rete tutti i nodi maliziosi una volta eseguite tutte le valutazioni del caso. Di certo non è in discussione l'utilità di un sistema di rivelazione di attacchi globale poiché si tratta pur sempre di uno strumento che fornisce agli utenti un certo grado di protezione dai cosiddetti attaccanti anche se in un contesto reale la sua incapacità o impossibilità di rimediare agli attacchi nel momento stesso in cui vengono eseguiti potrebbe sicuramente "intaccare" in maniera importante la sicurezza degli utenti della rete VANET.

La struttura dell'applicazione diventa sempre più completa ed allo stesso tempo complessa. L'implementazione di nuove funzionalità al suo interno, infatti, avviene mediante la definizione di nuovi *componenti* che richiedono però l'apporto di opportune modifiche alla sua composizione ai fini della loro esecuzione. Alle classi finora descritte sono state aggiunte una classe *Attacker* che modella un attaccante, una classe *Functionality* che raccoglie tutte le funzioni utili per lo svolgimento di specifiche operazioni e per la comunicazione con la base di dati, una classe *Correspondance* che modella una comunicazione tra due veicoli durante l'arco della simulazione definendo la relazione sorgente-destinatario, una classe *DetectSuppression* che implementa la strategia per il post-processing dei beacon ai fini della rilevazione di un attacco di Beacon Suppression e, per fnire, una classe *AnomalyLog*, che verrà ripresa e approfondita successivamente, utile per modellare le funzionalità ed il comportamento del cosiddetto Anomaly Checker.

6.3 Simulazione degli attacchi

Nel capitolo precedente è stato messo sotto la lente di ingrandimento il modello ideale della OBU con tutti i blocchi che ne indicano le componenti più interessanti (Figura 5.6) e sono state ampiamente valutate le vulnerablità di questo importantissimo sistema che consente la comunicazione e lo scambio di informazioni e dati tra veicoli attraverso l'ormai noto *processo di beaconing*. Questa parte del documento è interamente dedicata alla creazione ed esecuzione delle simulazioni degli attacchi che possono essere messi in atto sfruttando le suddette vulnerabilità dei vari elementi.

Di seguito verranno riprese tutte le varie tipologie di attacco già esposte nel Paragrafo 5.8 del Capitolo 5 e descritte le modalità con le quali è stato possibile definire ed eseguire la relativa simulazione. Il sistema di partenza sul quale verranno impostate le simulazioni è quello descritto nel Capitolo 5: il database e l'applicazione SBX verranno all'occorrenza modificati per poter consentire una corretta esecuzione delle simulazioni. Per ognuno dei componenti presenti all'interno della On Board Unit verranno valutate e simulate le diverse tipologie di attacco a cui può essere soggetto e per ogni possibile attacco la trattazione si compone di due parti:

- una prima parte che comprende la descrizione generale dell'attacco e dello scenario simulato e nella quale viene indicata una possibile stategia per la sua mitigazione;
- una seconda parte nella quale vengono raccolti e discussi i risultati dei test effettuati in seguito alla simulazione dell'attacco e dello schema adottato per tentare di smascherarlo e possibilmente combatterlo.

Poiché in questa fase dovranno essere definite le simulazioni per inscenare i vari tipi di attacco sulla rete VANET e poiché per far ciò è necessario creare delle funzioni ad hoc che svolgano determinate operazioni ho preferito organizzare al meglio la struttura interna dell'applicazione creando una nuova classe chiamata *Attacks* nella quale verranno inserite oltre agli attacchi veri e propri anche le strategie per la rispettiva mitigazione.

6.3.1 Considerazioni preliminari

Le applicazioni di sicurezza, come detto più volte, sono estremamente importanti se non addirittura indispensabili per una Vehicular Ad Hoc Network. Queste, a differenza delle applicazioni orientate all'infotainment, richiedono una latenza di trasmissione dei dati minima affinché una situazione di pericolo possa essere tempestivamente annunciata ai nodi interessati. Molti studi del settore hanno posto la loro attenzione sull'affidabilità e sulla qualità della comunicazione e delle informazioni trasmesse e scambiate dai nodi considerando in modo particolare la cosiddetta "latenza di trasmissione". Grazie ad un gran numero di test eseguiti su reti con configurazioni sempre diverse è stato possibile effettuare una stima di questo ritardo, talvolta chiamato semplicemente *latenza*, il quale specifica il tempo massimo che può intercorrere tra la creazione del beacon da parte del nodo sorgente e la rispettiva ricezione da parte del destinatario o dei destinatari senza che venga "intaccata" la qualità dei dati. Una volta nota la latenza massima che un beacon può "sopportare" durante l'inoltro sulla rete diventa semplice per i ricevitori valutarne la "freschezza" e l'attendibilità attraverso il calcolo del tempo trascorso dalla data di creazione a quella di ricezione.

Molti dei requisiti e dei *parametri* utili per la corretta configurazione di una rete VANET sono stati definiti da importantissimi progetti di ricerca in giro per il mondo come il VSC (*Vehicle Safety Communications*) in USA o il C2C-CC (*Car-to-Car Communication Consortium*) e l'ETSI TC ITS in Europa. Questi, in particolar modo, hanno provato che il valore massimo di latenza *concesso* ad un beacon affinché

non se ne causi l'invalidazione del contenuto è pari a 100 millisecondi [74]. Tuttavia è importante far notare che non sempre è possibile definire in modo preciso e soprattutto rispettare determinate condizioni quando la comunicazione avviene su un supporto intrinsecamente inaffidabile come nel caso qui considerato.

La trasmissione dei dati su qualunque canale e mezzo di comunicazione è soggetta a ritardi dovuti sia ai tempi propri dei sistemi per l'elaborazione e la creazione dei pacchetti che alle caratteristiche della rete. Un esempio può essere il tempo/ritardo di trasmissione che dipende dalla capacità del collegamento e dalla dimensione del pacchetto o il tempo/ritardo di propagazione di un pacchetto in un canale che risulta invece essere strettamente dipendente dalla distanza (d) che intercorre tra sorgente e destinatario e dalla velocità di trasmissione del mezzo (S). Il rapporto tra queste due grandezze ($^{d}/_{s}$) genera talvolta un valore talmente piccolo che nei casi migliori può addirittura essere trascurato. Nel complesso bisogna considerare diversi altri "fattori" oltre a quelli appena visti che sono sempre presenti e che quindi devono essere sempre considerati. Tra questi è importante citare lo stato del canale di comunicazione, la congestione della rete o eventuali errori di trasmissione che potrebbero senz'altro comportare l'incremento dell'entità del ritardo end-to-end complessivo. Naturalmente in un contesto reale tutte queste proprietà non possono essere ignorate anche perchè è noto che un segnale nell'aria può essere soggetto ad attenuazione a causa degli ostacoli incontrati e delle superfici con le quali deve per forza di cose scontrarsi a meno che non ci si trovi in un'area deserta nella quale non sono presenti edifici o altri elementi che ne possono ostacolare la propagazione.

L'applicazione SBX idealizza la trasmissione delle informazioni tra i nodi sul canale di comunicazione DSRC dedicato e non mette in atto alcun sistema di valutazione dello stato di degrado del segnale e del canale ma si limita a simulare la perdita dei dati attraverso l'implementazione di un sistema basato sulla "*probabilità di ricezione*" che risulta essere funzione della distanza presente tra i nodi che in ogni dato *timestep* intendono comunicare (vedi Paragrafo 5.9 del Capitolo 5).

Nelle sezioni successive verranno valutate diverse tipologie di attacco alcune delle quali avranno l'obiettivo di apportare un ritardo alla trasmissione dei beacon o manipolare le informazioni relative alla data di creazione del pacchetto. In questi casi pur non considerando, come detto prima, gli effetti dei ritardi sulla trasmissione dei beacon sarà sufficiente eseguire un confronto tra il timing virtuale della simulazione ed il valore del timestamp *manipolato* considerando al più una latenza di 100 millisecondi che può essere considerata il ritardo massimo che un messaggio "appartenente" alle applicazioni di sicurezza può registrare.

6.3.2 Attacchi al ricevitore GPS

6.3.2.1 GPS Blocking

Descrizione Questa tipologia di attacco viene messo in atto dagli utenti maliziosi per creare dei disturbi nelle comunicazioni con gli altri nodi della rete. In particolare il bloccaggio delle funzionalità del GPS causerà una mancanza delle informazioni relative alla posizione del veicolo sui beacon da esso generati.

Tuttavia non sarà difficile per il ricevitore o per i ricevitori individuare una criticità di questo tipo: il sistema di ricezione dei beacon posto all'interno delle OBU dovrà semplicemente valutare i dati sulla posizione del nodo sorgente per rilevare l'anomalia e procedere con lo scarto del messaggio stesso.

L'impostazione della simulazione ha richiesto una leggera modifica all'applicazione SBX ed in particolare alla classe Attacks. Qui è stata creata una nuova funzione chiamata *doGPSBlockingAttack()* che ha il compito di simulare questo attacco. Al suo interno è stato definito un vettore contenente tutti gli istanti temporali in cui si vuole che un attaccante agisca. Inoltre uno dei 4 parametri richiesti da questa funzione definisce l'intervallo temporale o la durata dell'attacco. Ad esempio se la durata dell'attacco deve essere pari a 30 secondi e l'attacco inizia al timestep 120 allora vorrà dire che nell'intervallo [120, 150] l'attaccante sferrerà l'attacco a tutti i nodi vicini. Oltre al valore appena considerato alla funzione viene passato anche il nome di due attaccanti che rappresentano i nodi prescelti per effettuare l'attacco vero e proprio nei vari istanti di tempo predefiniti.

Lanciando la simulazione e valutando lo "stato" della tabella che ospita i beacon sul database senza che venga attuato alcun meccanismo di rilevazione è possibile scorgere con una certa evidenza la presenza di questi dati "manipolati". Una volta giunti virtualmente a destinazione sarà sufficiente che il modulo addetto alla ricezione dei beacon esegua un semplice check dei valori relativi alla posizione della sorgente all'interno dell'apposito campo del messaggio per schivare questo tipo di attacco ed evitare strane situazioni nella comunicazione interveicolare.

Risultati e considerazioni La strategia utilizzata dal sistema di rilevazione creato per lo smascheramento del presente attacco viene implementata all'interno della funzione chiamata *tryToDetectGPSBlockingAttack()*. Essa agisce sul contenuto dei beacon ed in particolare sui campi relativi alla posizione di colui che li ha generati affinchè possa valutare se è in corso o meno un attacco di questo tipo ed in caso positivo decidere di non salvarlo *virtualmente* sulla memoria. Al fine di ottenere un riscontro certo del lavoro svolto da questa funzione-filtro è stata impostata una seconda funzione che si preoccupa della sola creazione del file di log sul quale verranno, al termine della simulazione, indicati i dati riguardanti il numero totale di pacchetti manipolati volutamente dall'attacco ed il numero totale di pacchetti manipolati correttamente rilevati. Questo importantissimo strumento tornerà utile alla fine dell'intero processo per trarre delle conclusioni e delle considerazioni sull'affidabilità del sistema di rilevazione qui esposto. In particolare, dopo avere eseguito una lunga serie di test è possibile affermare che il sistema proposto ha un indice di affidabilità altissimo che permette di combattere efficacemente questa tipologia di attacco e di evitare il logging della totalità dei messaggi appositamente manipolati attraverso la simulazione del suddetto attacco.

Parametri richiesti dalla doGPSBlockingAttack()

- Beacon da *manipolare*

- Identificativo di due attaccanti
- Durata dell'attacco per ognuno dei timestep specificati nella funzione

Configurazione attacco e risultati delle simulazioni
- Attaccanti utilizzati per l'attacco: *Audinot_7_0* e *Pertini_20_10*
- Durata di ogni attacco: 10 secondi
- Durata della simulazione virtuale: 300 secondi
- Numero di test eseguiti: 10
- Indice di affidabilità (media) del sistema di protezione: 100%

6.3.2.2 GPS Spoofing

Descrizione Lo scenario che evidenzia questa tipologia di attacco in corso sulla rete prevede l'invalidazione dei dati relativi alla posizione dei nodi maliziosi e quindi la trasmissione di questi beacon appositamente manipolati alle vittime presenti nell'intorno di quegli attaccanti che decidono di metterlo in atto. Poiché l'intenzione dell'attaccante è quella di disseminare informazioni false in giro per la rete, è indispensabile che i nodi autorizzati facciano una certa attenzione nell'accettare in modo incondizionato e senza alcuna precauzione dati provenienti da utenti che potrebbero talvolta rivelarsi dei malintenzionati. Per mettere in atto un attacco di questo genere mediante una simulazione l'idea è quella di agire sul blocco che si preoccupa all'interno dell'applicazione SBX della generazione dei beacon. La modifica della classe Attacks ha comportato, anche in questo caso, la definizione di una funzione chiamata *doGPSSpoofingAttack()* il cui compito è quello di alterare le informazioni prelevate dal segnale GPS mediante l'aggiunta di un certo valore Δ che possa indurre i ricevitori a credere che quel dato nodo si trovi in una posizione diversa da quella nella quale risiede realmente. L'aggiunta costante di questo Δ sui beacon fake genera dei veicoli fantasma nelle posizioni da esso segnalate ovvero veicoli che effettivamente non si trovano in quella posizione. L'attacco verrà inscenato dagli attaccanti prescelti in determinati intervalli di tempo. Il sistema di rilevazione per questo attacco prevede una valutazione della distanza che intercorre tra il sorgente e il destinatario a partire dalle coordinate prelevate da tutti beacon "catturati" dal blocco interno alla On Board Unit addetto alla loro ricezione. Purtroppo questo "semplice" controllo non assicura che tutti i messaggi "fake" possano essere correttamente catturati in quanto potrebbe accadere che, nonostante l'aggiunta del Δ, la posizione dell'attaccante ricada ancora all'interno dell'area indicata dallo standard. Infatti, se l'attaccante e la vittima dovessero nella realtà trovarsi a pochi metri di distanza l'uno dall'altro per l'intera durata dell'azione di attacco, l'aggiunta del Δ potrebbe essere del tutto insufficiente per permettere al sistema di controllo di individuare l'attacco e l'attaccante che quindi, in casi come questo, potrebbe farla franca senza essere mai scoperto, a patto che il Δ non sia tanto elevato da "spostare" la vittima al di fuori dell'area di ricezione. La strategia di protezione pensata per combattere questa tipologia di attacco si appoggia anche alla cosiddetta lista nera: ogni nodo che individua un attaccante ne aggiunge l'identità all'interno

della blacklist locale cosicché alla ricezione di ogni nuovo beacon sarà sufficiente valutare dapprima la presenza dell'identità del sorgente sulla propria lista e poi, in caso questo dia esito negativo, procedere con la verifica della distanza. Naturalmente se il primo dei due controlli appena indicati dovesse dare esito positivo si potrà procedere direttamente con lo scarto, viceversa si procederà col secondo controllo e se anche questo dovesse dare esito negativo allora si potrà eseguire il salvataggio sulla propria memoria. Talvolta però, in un contesto reale, un nodo potrebbe essere obbligato a trasmettere involontariamente dei dati errati in seguito ad un problema di ricezione del segnale GPS o di trasmissione delle informazioni sul canale DSRC. E' in casi come questo che viene messo in evidenza il principale limite di questa strategia: un nodo "segnalato" non potrà mai più essere rivalutato da coloro i quali lo hanno etichettato come un possibile *attacker*, a meno che (nella realtà) non sia presente una terza parte fidata che a intervalli più o meno regolari distribuisca in broadcast agli utenti di ogni area della rete sia la lista di tutti coloro i quali sono stati erroneamente segnalati come degli attaccanti e sia la lista di coloro i quali invece sono stati effettivamente definiti attaccanti a seguito di opportune valutazioni e verifiche da parte degli organi preposti.

Oltre alla modalità di attacco appena descritta, si potrebbe pensare di definire uno schema alternativo con il quale un attaccante possa inscenare un attacco di GPS Spoofing. La strategia di attacco appena accennata consiste nell'utilizzo, da parte dell'attaccante, di una traccia alternativa riferita ad esempio al percorso seguito in un istante di tempo precedente da un altro veicolo o dallo stesso veicolo attaccante. Questa traccia, che sarà data in pasto alla funzione *doGPSSpoofingAttack()*, permetterà all'attaccante di generare messaggi a partire da una posizione fittizia totalmente differente da quella reale nella quale si trova e dalla quale riceve e trasmette i beacon. Il sistema di protezione utilizzato per lo smascheramento di quest'azione di attacco rimane inalterato in quanto la sua efficacia non risente del "cambio di strategia" a patto che l'attaccante non utilizzi una traccia di un veicolo molto vicino alla vittima. In questi casi, infatti, verrebbe a ripresentarsi il problema già discusso in precedenza della *vicinanza* vittima-attaccante che non permetterebbe, a seguito del controllo della distanza, di smascherare l'azione di attacco. Anche in questo contesto la presenza della blacklist rappresenta un plus per ogni nodo in quanto permette la definizione di una doppia linea di difesa che gli consentirà di difendersi con maggiore incisività grazie al bloccaggio di tutti i messaggi provenienti dagli utenti già presenti al suo interno.

Risultati e considerazioni Il sistema di rilevazione implementato attraverso la funzione *tryToBlockGPSSpoofingAttack()*, già ampiamente anticipato nella sezione precedente, basa il suo funzionamento sul controllo della distanza tra il destinatario di ogni beacon e la sorgente che nel peggiore dei casi potrebbe rivelarsi un attaccante. Lo spoofing consiste nella manipolazione dei dati relativi alla posizione del nodo sorgente all'interno del beacon e induce le vittime inconsapevoli a credere che un certo veicolo si trovi in un certo istante in una data posizione. La creazione di una posizione fittizia per un veicolo attaccante avviene mediante l'aggiunta di un offset alle coordinate spaziali

che, grazie ad un opportuno calcolo matematico, permette di "allontanare" la vittima da quest'ultimo in modo tale da creare l'illusione della distanza. L'offset o Δ può essere scelto liberamente ed in particolare nei test eseguiti si è deciso di impostarlo a circa 100 metri. Purtroppo, com'è facile intuire, non tutte le anomalie potranno essere correttamente rilevate dal sistema sviluppato poiché, come già messo in evidenza in precedenza, nonostante l'aggiunta dell'errore potrebbe capitare che l'attaccante continui a risiedere nell'intorno della vittima e quindi nell'area indicata dallo standard obbligando la vittima a doverne loggare tutti i messaggi. La presenza della lista nera risulta particolarmente efficace per il contesto d'utilizzo: essa infatti essendo costantemente aggiornata con le informazioni degli attaccanti via via individuati permetterà al sistema di scartare tutti i messaggi anche non manipolati provenienti da una certa sorgente anche se rientrano all'interno dell'area definita dal range di trasmissione. In conclusione, grazie ai numerosi test eseguiti è stato possibile valutare lo schema che ha dimostrato la sua ottima capacità ed efficacia ottenendo un indice di affidabilità medio pari a 92,7% nel caso in cui Δ=100 metri. Naturalmente molto dipende dal valore del Δ scelto: se questo dovesse essere molto piccolo, diciamo qualche metro, allora sarà veramente difficile per qualsiasi sistema riuscire a smascherare l'attacco senza far affidamento a strumenti diversi dai soli dati presenti sui beacon e da quelli prelevati dai ricevitori a bordo delle OBU.

Parametri richiesti dalla doGPSSpoofingAttack()

- Beacon da *manipolare*
- Identificativo di tre attaccanti
- Posizione geografica del veicolo "vittima"
- Offset o Δ

Configurazione attacco e risultati delle simulazioni

- Attaccanti utilizzati per l'attacco: *Gandhi_40_0*, *Gandhi_60_18* e *Malvasia_94_8*
- Durata della simulazione virtuale: 300 secondi
- Δ = 100 metri
- Numero di test eseguiti: 10
- Indice di affidabilità (media) del sistema di protezione: 92.7%

6.3.3 Attacchi al generatore di beacon

6.3.3.1 Fake data

Descrizione Uno dei possibili attacchi che riguardano questo importante componente della OBU consiste nella manipolazione dei dati che compongono i beacon, come le coordinate geografiche, il timestamp o le identità, al fine di arrecare dei fastidi ai nodi autorizzati durante la comunicazione interveicolare. Per realizzare uno scenario virtuale che possa simulare questa tipologia di attacco è stata sviluppata una nuova funzione chiamata *doFakeBeaconsAttack()*. Essa

è composta da due vettori che contengono rispettivamente gli istanti temporali in cui i nodi malintenzionati devono mettere in atto l'azione di disturbo e una serie di identità predefinite che possono essere usate dall'aggressore durante l'esecuzione dell'attacco per variare il proprio identificativo. Tra i parametri richiesti da questa funzione è possibile trovare: due valori utili rispettivamente per la falsificazione dei dati sulla posizione e sul timestamping all'interno del beacon, un intero che definisce la durata dell'attacco e gli identificativi dei nodi malintenzionati che nei vari istanti di tempo dovranno sferrare l'attacco alle vittime inconsapevoli.

Il punto è: l'attaccante deve cambiare tutti i valori ogni volta oppure può decidere di alterarne uno piuttosto che un altro? Dal mio punto di vista, al fine di rendere l'attacco ancora più imprevedibile e difficile da rilevare e combattere, sarebbe opportuno che ogni attaccante utilizzi uno schema variabile. Durante l'operazione ogni attaccante potrà decidere quali dei tre parametri modificare. Questo "comportamento" aleatorio potrà senza dubbio rappresentare un'arma in più per l'attaccante e un ulteriore ostacolo per la vittima al fine dell'eventuale rilevazione che potrà essere messa in atto solo a seguito di una oculata ispezione del contenuto dei messaggi in ingresso. Tuttavia vi sono delle situazioni in cui diventa quasi impossibile per la vittima rilevare l'anomalia. Basti pensare ai casi in cui un malintenzionato nonostante il cambio della propria identità e/o posizione dovesse rimanere all'interno dell'area definita dallo standard. Lo scenario invece varia se una volta falsificata l'informazione sulla posizione il nodo risieda al di fuori dell'area indicata dallo standard oppure se viene semplicemente alterato il timestamp. Il cambio di identità potrebbe essere difficile da individuare a meno che un attaccante stolto non vada ad utilizzare i dati identificativi di qualche altro nodo che risiede come lui nello stesso intorno della vittima. In questo caso infatti la vittima ispezionando i dati sulla posizione dei due potrà rilevare una certa difformità che lo porterà ad intuire la presenza di una situazione strana e quindi di una possibile anomalia. Anche in questo caso si potrebbe adottare la strategia già vista in precedenza della "*lista nera*": se un nodo dovesse rilevare un'anomalia allora viene salvata l'identità di colui che l'ha generata su una lista locale in modo che se una situazione simile dovesse ripresentarsi una seconda volta esso potrà difendersi attraverso un rifiuto persistente dei messaggi provenienti da quella data sorgente.

Consideriamo adesso la definizione di una possibile strategia di rilevazione e prevenzione per questa tipologia di attacco procedendo per gradi e cercando di coprire tutte le possibili casistiche.

1. **Falsificazione della posizione**
   Questa "situazione" può essere individuata dal nodo vittima soltanto mediante la valutazione dei campi relativi alla posizione della sorgente all'interno del beacon ricevuto. Essa può in un certo senso essere ricondotta al caso già trattato e valutato del GPS Spoofing: sia le considerazioni generali che le tecniche utili per una eventuale mitigazione corrispondono esattamente a quelle già viste nel caso del GPS Spoofing.

2. **Falsificazione del timestamp**
   L'obiettivo dell'attaccante è quello di fuorviare il nodo vittima attraverso la generazione e la successiva trasmissione di informazioni volutamente manipolate che potenzialmente potrebbero far riferimento a situazione già avvenute nel passato. L'attacco consiste nell'associazione di dati fittizi al campo *timestamp* del beacon. Anche in questo caso l'unico metodo utile per una possibile valutazione della qualità dei dati ricevuti consiste nella verifica delle proprietà dei beacon ed in particolare del campo contenente l'informazione relativa alla loro data di creazione.

   Ma un nodo come può conoscere l'informazione sul "tempo" in modo sicuro per poter decidere se un beacon è attuale o meno? In un contesto reale il ricevitore dei beacon potrebbe affidarsi ad un clock appositamente installato sulla OBU oppure all'informazione sul tempo trasportata dal segnale GPS. Nell'ambiente virtuale invece l'unico metodo possibile e certo consiste nella lettura dei dati relativi al timing della simulazione generati dal simulatore SUMO.

   Naturalmente la strategia per la rilevazione di questa tipologia di attacchi deve basarsi sulla valutazione della data di creazione del pacchetto: se al momento della ricezione questa dovesse riferirsi ad un istante temporale più lontano rispetto ai 100 millisecondi indicati dal requisito massimo di latenza allora si procede con la segnalazione ed il relativo scarto del messaggio ricevuto.

3. **Falsificazione dell'identità**
   La continua variazione dell'identità di un nodo potrebbe indurre gli altri nodi che riceveranno i beacon con questi dati falsati a pensare che il proprio vicinato sia composto, in un certo intervallo di tempo, da un numero di veicoli superiore rispetto a quello reale. Un sistema di rilevazione per questa tipologia di attacco prevede la valutazione di tutti i beacon ricevuti in ogni timestep ed in particolare dei dati relativi all'identità della sorgente. Purtroppo non sempre sarà possibile riuscire a scoprire un mascheramento da parte di un attaccante in quanto basandoci solo sul dato relativo all'identità dei nodi non vi è modo per capire in tempo reale se colui che in un certo istante t trasmette un dato con l'identità X è lo stesso di quello che lo trasmette all'istante $t+\Delta$ con l'identità Y. L'unico caso in cui è possibile smascherare un nodo che inscena un attacco di questo genere con il solo dato sull'identità come mezzo di paragone è quando questo inavvertitamente utilizza un identificativo appartenente ad un altro nodo autorizzato che però risiede nella stessa area: in questo caso una vittima si ritroverebbe a dover loggare due messaggi appartenenti apparentemente alla stessa entità ma provenienti da posizioni diverse nello stesso istante temporale. Nella fattispecie diventa facile per la vittima individuare l'anomalia anche se risulta piuttosto difficile riuscire a definire una possibile "punizione" per evitare in futuro di esserne colpito nuovamente. Infatti se un nodo decidesse di bloccare in ricezione i dati provenienti da una certa sorgente ritenuta "fake" potrebbe allo stesso tempo andare a bloccare un altro nodo vittima come lui di un attacco a cui è stata però sottratta l'identità.

Risultati e considerazioni La strategia messa in atto attraverso la funzione *tryToBlockFakeBeaconAttack()* per rendere possibile l'individuazione di un attacco di questo tipo in corso sulla rete si basa sulla verifica della veridicità e della coerenza dei dati immessi dai nodi sorgente all'interno dei beacon trasmessi ed in particolare è composta da:

- un controllo sulla posizione del mittente basato sul calcolo della distanza sorgente-destinatario;
- un controllo sull'attualità dei dati basato sulla verifica del timestamp presente all'interno del beacon (in un contesto reale l'ideale sarebbe disporre di un clock locale che consenta al sistema di rilevazione di accedere in maniera sicura all'informazione sul tempo mentre nell'ambiente simulato, poichè non si dispone di questa entità aggiuntiva, è stato utilizzato come dato di confronto "sicuro" l'informazione generata da SUMO e presente all'interno del campo *timestep* della traccia che ha dato vita a quel beacon);
- un controllo sull'identità della sorgente inserita nel campo *id_src* del beacon basato sull'utilizzo di una sorta di blacklist che viene "creata" durante il corso della simulazione contenente le identità di tutti i nodi maliziosi che vengono via via smascherati.

Il terzo dei controlli appena descritti mette in risalto un'importante debolezza del sistema di rilevazione in quanto se da un lato grazie alla presenza della blacklist è possibile rifiutare tutti i dati provenienti da un certo utente che è stato etichettato come *attaccante* dall'altro la sua utilità potrebbe venire meno nell'ipotesi in cui un attaccante continui ad inventare e ad inserire nei beacon identità sempre diverse.

L'esecuzione dei test ha permesso di valutare la credibilità del sistema sviluppato ed in particolare ha decretato la sua ottima capacità nel rilevare i dati e i nodi maliziosi permettendo inoltre di evitare il logging di tutti quei pacchetti che risultano provenire da fonti corrotte anche al di fuori degli intervalli temporali in cui si è deciso di simulare l'attacco. Naturalmente potranno presentarsi delle situazioni in cui un attaccante potrebbe non essere "riconosciuto" e fermato dalla vittima. Uno di questi casi corrisponde alla prima trasmissione di dati da parte di un attaccante: la blacklist non avendo alcuna infiormazione al suo interno relativa a quel dato nodo non potrà in nessun modo permettere al sistema di smascherare l'attacco e l'attaccante che verrà sicuramente "segnalato" non appena verrà beccato a trasmettere dati falsi sulla propria posizione o sulla "freschezza" delle informazioni immesse sulla rete. Il punto debole del sistema di rilevazione, come già anticipato, è relativo al caso in cui un attaccante modifichi esclusivamente la sua identità fingendosi un altro. Poichè esso può fare affidamento soltanto alle informazioni presenti sui beacon e non ad elementi "esterni" per la sicurezza della rete che possano permettere di individuare l'anomalia in maniera diretta mediante la verifica delle credenziali o del certificato rilasciato da una terza parte fidata è normale aspettarsi che non tutte le operazioni di attacco possano essere correttamente rilevate e mitigate. Dai test eseguiti è emerso esattamente quanto appena detto: a differenza di

altre situazioni in cui sono stati ottenuti degli indici altissimi che indicavano una protezione totale della rete dall'attacco considerato, nel caso qui trattato ci si deve "accontentare" di un valore poco superiore al 90%.

Parametri richiesti dalla doFakeBeaconsAttack()

- Beacon da *manipolare*
- Identificativo di tre attaccanti
- Posizione geografica del veicolo "vittima"
- Durata dell'attacco per ognuno dei timestep specificati nella funzione
- Ritardo (in secondi) da "attribuire" alla data di creazione reale del beacon
- Distanza (in metri) da "aggiungere" a quella reale per la definizione delle nuove coordinate

Configurazione attacco e risultati delle simulazioni

- Attaccanti utilizzati per l'attacco: *Audinot_7_0*, *Gandhi_60_18* e *XXI_Aprile_12_10*
- Durata di ogni attacco: 20 secondi
- Ritardo: 0.5 secondi
- Distanza aggiuntiva: 80 metri
- Durata della simulazione virtuale: 300 secondi
- Numero di test eseguiti: 10
- Indice di affidabilità (media) del sistema di protezione: 91.3% (in 2 dei test eseguiti il sistema con l'aiuto della blacklist è riuscito a bloccare più messaggi manipolati di quelli realmente creati)

6.3.3.2 Active flooding

Descrizione Questo attacco consiste nell'invio massivo di beacon da parte dei cosiddetti attaccanti i quali, non rispettando la frequenza di trasmissione indicata dallo standard, tenteranno di sovraccaricare i canali DSRC e la rete al fine di limitarne la capacità di trasmissione e causare un calo delle prestazioni e del throughput per interferire nella comunicazione tra i nodi autorizzati e generare dei grossi disagi alle applicazioni di sicurezza. L'active flooding può essere a tutti gli effetti considerato un attacco DoS e se ad inscenarlo fossero un insieme di daemon coordinati nell'azione di disturbo allora l'obiettivo sarà più facile da raggiungere grazie ad una maggiore capacità distruttiva.

La simulazione di questo attacco prevede l'utilizzo della nuova funzione *doActiveFloodingAttack()* definita all'interno della classe Attacks la quale, a partire dalle tracce generate da SUMO, crea e trasmette virtualmente i pacchetti duplicati sulla rete la cui quantità unitamente all'identificativo dell'attaccante, alla durata dell'attacco e al beacon da replicare, rappresenta i parametri richiesti dalla suddetta funzione per poter svolgere il suo "lavoro".

La cattiva condotta dei nodi maliziosi potrà essere individuata dagli utenti autorizzati soltanto previa valutazione dei messaggi in ingresso alla OBU e, poiché trattasi di pacchetti duplicati, il sistema di rilevazione dovrebbe valutare in ogni timestep la presenza di copie multiple di un certo beacon ed

allo stesso tempo indicare una procedura che consenta di sopprimere coloro i quali stanno agendo in modo inappropriato. Un sistema di rilevazione che non preveda la comunicazione di un'anomalia ad una terza parte fidata in una situazione di attacco risulterebbe quasi inutile in quanto una certa sorgente che potrebbe essere stata bloccata localmente da uno o più nodi potrebbe in realtà continuare indisturbata con la sua azione di attacco alla rete. Idealmente quindi, a seguito della rilevazione di un comportamento inadeguato, un nodo dovrebbe attuare un certo protocollo che indichi i passaggi da seguire per poter comunicare a chi di competenza l'anomalia per cercare di "ripulire" la rete ed estromettere tutti quei nodi che causano queste pericolose situazioni di disturbo ad una rete prevalentemente orientata alla sicurezza degli utenti.

Dal momento che un attaccante "serio" che intenda attuare questo tipo di attacco può decidere di trasmettere massivamente i messaggi utilizzando di volta in volta degli identificativi diversi, ne consegue che un sistema di filtraggio altrettanto "serio" in ingresso al ricevitore dei beacon non può semplicemente affidarsi al controllo dell'identità della sorgente per rintracciare eventuali attacchi di active flooding. In ragione di ciò si potrebbe decidere di eseguire un controllo aggiuntivo sulla posizione dei nodi in quanto è pressoché impossibile che due veicoli con id diversi si trovino esattamente nella stessa posizione (ammesso e non concesso che i dati rilevati dai segnali GPS abbiano una certa precisione) anche se, come già mostrato, i dati relativi alla posizione possono anch'essi essere facilmente manomessi dagli attaccanti. Detto ciò, in un contesto reale, una possibile soluzione per smascherare questo tipo di attacco potrebbe consistere nella presenza di una terza parte fidata che, quando chiamata in causa, possa accedere liberamente ai dati prelevati dai sensori e dai radar a bordo dei veicoli, ai messaggi incriminati ricevuti dai vari nodi e alle immagini prelevate dalle telecamere poste talvolta in concomitanza delle RSU in modo tale da poter condurre un'analisi più approfondita e attraverso una loro interpolazione cercare di individuare il veicolo che ha compiuto l'atto illecito al fine di consentire alle autorità competenti di sottoporlo alle giuste conseguenze.

Risultati e considerazioni Il sistema sviluppato per ovviare ad attacchi di questo genere sulla rete virtuale fa affidamento ad una "strategia di ricezione dei beacon" che simula l'arrivo istante per istante dei pacchetti e ne esegue la verifica per individuare eventuali duplicati affinché se ne possa evitare il salvataggio in memoria. Ogni pacchetto ricevuto dal nodo viene inserito all'interno di una lista associata ad una HashMap la cui chiave è rappresentata dall'identificativo dello stesso nodo destinatario. In ogni timestep e per ogni beacon ricevuto è sufficiente effettuare un controllo del contenuto del "buffer di ricezione" associato al nodo ricevitore per decidere il destino del messaggio: il sistema inserisce "momentaneamente" tutti i pacchetti ricevuti in un dato istante temporale all'interno del buffer per eseguirne la verifica e se dovesse riscontrare dei duplicati allora provvederà immediatamente alla loro eliminazione. Affinché il sistema possa essere reso efficiente si procede ad intervalli regolari con la "pulizia del buffer di ricezione" in modo da fare un corretto utilizzo della memoria a disposizione durante la simulazione.

La tecnica appena presentata viene implementata all'interno della funzione *tryToBlockActiveFloodingAttack()* e durante i test è sempre risultata molto efficace tanto da riuscire costantemente a rilevare le criticità appositamente create ottenendo un altissimo indice di affidabilità.

Parametri richiesti dalla doActiveFloodingAttack()
- Beacon da *manipolare*
- Identificativo di un attaccante
- Durata dell'attacco per ognuno dei timestep specificati nella funzione
- Numero di copie da replicare di ogni beacon manipolato

Configurazione attacco e risultati delle simulazioni
- Attaccanti utilizzati per l'attacco: *Audinot_7_0*
- Durata di ogni attacco: 2 secondi
- Numero di repliche del beacon: 10
- Durata della simulazione virtuale: 300 secondi
- Numero di test eseguiti: 10
- Indice di affidabilità (media) del sistema di protezione: 100%

6.3.4 Attacchi al trasmettitore di beacon

6.3.4.1 Beacon suppression

Descrizione Il trasmettitore dei beacon è un modulo interno alla OBU orientato alla trasmissione di tutti i messaggi *elaborati* dal generatore di beacon. Un attaccante potrebbe voler manipolare questo modulo per limitarne le funzionalità e causare degli inconvenienti durante la comunicazione interveicolare andando, ad esempio, a "sopprimere" tutti i beacon che possiedono determinate proprietà. Un malintenzionato potrebbe inoltre decidere di bloccare la trasmissione dei beacon totalmente oppure parzialmente in seguito al verificarsi di particolari situazioni, come nei casi in cui questo stia percorrendo una strada ad una velocità maggiore rispetto a quella consentita.

La simulazione di questo attacco avviene grazie alla funzione *doBeaconSuppressionAttack()* presente all'interno della classe Attacks. Essa, oltre al beacon da "verificare", riceve come parametri due valori interi che rispettivamente indicano la modalità e la durata dell'attacco, gli identificativi degli attaccanti e, opzionalmente, gli identificativi delle vittime e la velocità massima consentita al veicolo attaccante. Le modalità di attacco prima citate sono le seguenti tre:

1. soppressione totale dei beacon in uscita dal nodo attaccante (modo 1);
2. soppressione dei soli beacon aventi una certa destinazione sulla rete (modo 2);
3. soppressione di tutti i beacon nei casi in cui l'attaccante stia procedendo ad una velocità maggiore ai 50 km/h consentiti dal codice della strada per i centri urbani come nel caso dello scenario utilizzato per la simulazione (modo 3).

Seguendo queste strategie durante la fase di beaconing sarà possibile selezionare i messaggi "incriminati" al fine di evitarne il salvataggio sulla tabella della base di dati che simula la memoria delle OBU dei veicoli destinatari.

Un sistema di rilevazione valido per lo smascheramento di questa tipologia di attacco può semplicemente basarsi su un "servizio" ad hoc che esegua una analisi dei dati ricevuti ogni n timestep per individuare eventuali situazioni che possano far presagire la presenza di un attacco di questo tipo in corso. Esso in pratica dovrà valutare il vicinato del nodo in esame e verificare che non vi siano dei veicoli che continuamente compaiono e scompaiono, proprio come dei fantasmi. Nel momento in cui un nodo autorizzato rileva un comportamento di questo genere da parte di un possibile attaccante, il suddetto servizio dovrebbe attuare un protocollo che *idealmente* potrebbe consistere nell'aggiunta immediata dell'identità del nodo incriminato su una blacklist e nel rifiuto di tutti i dati provenienti da esso oppure nella comunicazione dell'avvenuta rilevazione dell'attacco ad una terza parte fidata.

Risultati e considerazioni La simulazione dell'attacco di Beacon Suppression prevede l'inibizione del trasmettitore di beacon sulla On Board Unit del nodo attaccante e quindi l'assenza di messaggi da esso generati sulle memorie dei nodi vicini. Poichè l'attacco riguarda il solo dispositivo di trasmissione è normale che l'attaccante continui a ricevere messaggi negli stessi momenti in cui sta inscenando l'attacco e proprio su questa "particolarità" si basa la strategia di rivelazione sviluppata. In particolare, questa fonda il suo funzionamento sul concetto di differenza tra numero di pacchetti trasmessi e numero di pacchetti ricevuti da ognuno dei veicoli che prendono parte alla simulazione, anche se ritiene necessario dover accedere alla totalità dei beacon scambiati. Da quanto detto è facile dedurre che il presente sistema di protezione è un sistema globale e *offline* che non permette una *rilevazione al volo* come avviene nei casi finora trattati. Questi, grazie a dei controlli e a delle verifiche di coerenza delle informazioni presenti sui campi dei beacon, permettono di individuare a caldo un beacon "fake" e quindi la possibile presenza di un attacco in corso che così facendo può essere facilmente contrastato mediante il blocco locale dei dati provenienti dai nodi maliziosi via via individuati e l'immediata comunicazione dell'anomalia rilevata ad una terza parte fidata che effettuando i dovuti controlli potrebbe poi decidere di procedere con la loro espulsione dalla rete.

Il sistema implementato per lo smascheramento di un attacco di beacon suppression deve affidarsi ad una sorta di post-processing dei dati memorizzati sulla base di dati. Esso, oltre all'accesso alla totalità dei beacon, ritiene indispensabile il prelievo dei dati presenti sulla tabella *correspondance* sulla quale, durante la fase di beaconing, vengono memorizzate tutte le corrispondenze ovvero tutte le possibili coppie di veicoli che nel corso della simulazione si sono scambiati almeno una volta delle informazioni. Questa importantissima tabella, che al termine della simulazione conterrà miglialia di righe e sulla quale vengono inserite in modo esclusivo e senza ripetizioni tutte le possibili "corrispondenze", sarà acceduta e valutata durante la fase di rilevazione dell'attacco attraverso una strategia implementata all'interno di una classe opportuna chiamata *DetectSuppression* che verrà eseguita in un ambiente multi-thread al

fine del miglioramento delle prestazioni generali dell'applicazione. La suddetta strategia di protezione può essere brevemente descritta attraverso i seguenti passaggi:

1. prelievo di un blocco di dati dalla tabella *corripondance*;
2. prelievo degli identificativi dei nodi da ogni corrispondenza;
3. calcolo del numero di pacchetti inviati dal *sorgente* durante l'intera simulazione al *destinatario*;
4. calcolo del numero di pacchetti inviati dal *destinatario* durante l'intera simulazione al *sorgente*;
5. calcolo della differenza tra i valori calcolati ai punti 3 e 4 per valutare una eventuale discordanza tra dati trasmessi e ricevuti;
6. valutazione finale del valore ottenuto al punto 5 ed eventuale inserimento dell'identificativo nella lista dei possibili attaccanti.

Purtroppo il sistema di logging implementato non è di particolare aiuto per la strategia appena descritta in quanto basandosi sul concetto di "probabilità di trasmissione" (vedi paragrafo 5.9 del precedente capitolo) il numero di pacchetti scambiati da due nodi limitrofi in un certo `timestep` potrebbe non corrispondere causando quindi dei fastidi durante l'esecuzione dei calcoli prima indicati. Durante la fase di sviluppo e testing del sistema di protezione infatti è capitato molto spesso di vedere delle differenze di questo tipo e per tale motivo si è deciso di procedere con il calcolo e la definizione di un nuovo valore che indica il numero medio di pacchetti che possono essere andati persi durante l'intero arco della simulazione tra ogni coppia di nodi che almeno una volta si sono scambiati dei messaggi. L'idea è che gli attaccanti inibendo le funzionalità del trasmettitore di beacon trasmetteranno meno dati rispetto a quelli ricevuti e di contro i destinatari di questi pacchetti non trasmessi si ritroveranno a dover trasmettere più dati di quelli effettivamente ricevuti. Poichè la differenza tra i due valori per un attaccante sarà consistente, sarà piuttosto semplice procedere con l'identificazione dei cosiddetti utenti "buoni" che invece registreranno per ogni comunicazione instaurata con un dato nodo una differenza tra pacchetti trasmessi e ricevuti molto limitata se non nulla (caso ottimo in cui tutti i pacchetti sono correttamente ricevuti e correttamente trasmessi).

Durante i test il sistema ha mostrato la sua efficacia in tutte e tre le modalità di attacco riuscendo a identificare con successo *anche* i nodi maliziosi di volta in volta passati alla *doBeaconSuppressionAttack()* per simulare l'attacco. Tuttavia, a causa dell'aleatorietà del sistema di trasmissione implementato, potrebbe talvolta essere individuato e mostrato l'identificativo di un nodo buono che per ragione diverse abbia magari superato la famosa soglia della media prima calcolata. Per concludere quindi è possibile asserire che i test eseguiti hanno consentito di provare la qualità e l'affidabilità del sistema sviluppato che pertanto si è dimostrato uno strumento valido per consentire l'individuazione dei nodi che sono stati prescelti per l'esecuzione dell'attacco di beacon suppression. Essa non è stata implementata mediante una funzione come in

tutti gli altri casi ma attraverso una classe vera e propria che venendo lanciata in un ambiente multi-thread effettua una valutazione parallelizzata dei dati prelevati timestep per timestep e fornisce in output un resoconto all'interno di un file di log sul quale vengono listate le identità dei nodi che si sono rivelati dei *potenziali* attaccanti.

Il sistema di protezione appena descritto può andare bene fintantoché l'attaccante non decide di interdire anche il funzionamento del chip interno alla OBU che si occupa della ricezione dei beacon. Come specificato inizialmente, lo schema di difesa implementato si appoggia al concetto di "differenza tra dati inviati e dati ricevuti" da ogni nodo sulla rete. Bloccando contemporaneamente la ricezione dei beacon un attaccante potrebbe di conseguenza raggirare il sistema di rilevazione e non essere mai identificato in quanto l'esito restituito dal suddetto controllo risulterà *nullo*. Onde evitare una situazione del genere si è preferito procedere con lo sviluppo e la definizione di uno schema alternativo che potesse sopperire ai limiti del sistema precedente. L'idea che sta alla base del nuovo algoritmo di difesa è stavolta connessa al concetto di *spostamento* dei veicoli. Questo, per ogni coppia di veicoli presenti all'interno della tabella *correspondance*, proverà a rilevare gli intervalli temporali in cui non è avvenuto lo scambio di dati. I motivi per cui due nodi potrebbero non aver scambiato dei dati per un certo lasso di tempo possono essere molteplici. Un possibile esempio potrebbe essere rappresentato da uno scenario ipotetico in cui il sorgente potrebbe essersi fermato a fare rifornimento raggiungendo nuovamente il destinatario in coda ad un incrocio e riprendendo quindi la trasmissione di beacon. Tuttavia un nodo che scompare completamente dalla rete in un certo istante di tempo (non sarà presente alcun beacon trasmesso da quel nodo all'interno dell'area nella quale risiedeva) e ricompare n istanti di tempo dopo in una posizione totalmente diversa da quella precedente fa presagire la presenza di qualche stranezza sulla rete. Il nodo in questione, che potrebbe quasi sicuramente essere un malintenzionato, dovrà dimostrare alle autorità competenti quale sia stata la ragione per la quale si sia verificata quella discrepanza e, a meno che non dimostri di essere stato spostato da un mezzo di soccorso a causa di un guasto meccanico, esso potrà essere facilmente etichettato come attaccante e potrà essere di conseguenza sottoposto alle "punizioni" più appropriate. La strategia sviluppata fa leva proprio sulle considerazioni appena fatte. Tuttavia alcuni studi hanno dimostrato che un nodo che non trasmette alcuna informazione per un intervallo massimo di 3 secondi non può essere ritenuto nocivo per la rete in quanto non provocherebbe alcun danno agli eventuali sistemi che, sulla base dei dati scambiati dai nodi, effettuano la ricostruzione degli scenari di traffico per la valutazione ad esempio di incidenti o di situazioni particolari avvenute sulla rete stradale. Di contro ci sarebbe da discutere sul fatto che quell'utente si possa trasformare in questi casi in un fantasma che risulterà temporaneamente "invisibile al sistema logging" e che potrebbe quindi essere potenzialmente la causa di incidenti. Prendendo quindi per buoni i dati ottenuti da studi precedenti, durante l'esecuzione dei controlli, tutti i veicoli che non trasmetteranno alcun beacon per un periodo pari al più a 3 secondi non verranno considerati come possibili attaccanti. Unitamente al controllo appena indicato verrà effettuato un check sulla posizione

del nodo per valutare se questo si sia spostato durante l'arco di tempo in cui è *scomparso dai radar*. Ogni qual volta verrà riscontrata un'anomalia su un certo nodo dai controlli appena descritti si procederà con la valutazione dello "stato di trasmissione globale" attraverso il quale sarà possibile verificare se durante l'intervallo rilevato esso abbia trasmesso dati ai propri vicini. Nel caso di *total suppression* questo valore sarà sicuramente nullo e permetterà quindi di stabilire che quello effettivamente potrebbe essere un attaccante. Il limite di quest'altra strategia viene a palesarsi nel momento in cui un attaccante decide di non trasmettere delle informazioni *esclusivamente* ad un sottoinsieme specifico di nodi. In questi casi infatti il controllo globale sui dati trasmessi sarebbe del tutto vano e non consentirebbe quindi di stanare l'attaccante dal momento che il calcolo sul numero di beacon trasmessi ritornerà quasi certamente un valore diverso da zero.

Volendo tirare le somme si può affermare che in fin dei conti questo schema di rilevazione *globale* funziona abbastanza bene e, al di là del limite appena indicato, riesce a svolgere con una certa efficacia il suo lavoro. Come avveniva nel caso precedente, la lista dei nodi indicata all'interno del file di log e salvata sulla tabella Anomaly della base di dati sarà semplicemente una lista dei *probabili attaccanti* la quale potrà eventualmente rappresentare un punto di partenza per lo svolgimento di un'analisi ancora più approfondita nell'obiettivo di poter decretare con certezza assoluta le identità dei veri attaccanti.

Parametri richiesti dalla doBeaconSuppressionAttack()

- Beacon da *manipolare*
- Identificativo di due attaccanti
- Identificativo di due vittime
- Durata dell'attacco per ognuno dei timestep specificati nella funzione
- Modalità di funzionamento dell'attacco
- Velocità massima per un attaccante che una volta oltrepassata inibisce il trasmettitore di beacon

Configurazione attacco e risultati delle simulazioni

- Attaccanti utilizzati per l'attacco: *Pertini_20_10* e *XXI_Aprile_12_10*
- Velocità massima: 50 km/h
- Durata di ogni attacco: 30 secondi
- Durata della simulazione virtuale: 300 secondi
- Numero di test eseguiti: 10
- Il sistema di detection sviluppato permette di individuare correttamente i nodi scelti per l'esecuzione dell'attacco. Tuttavia è doveroso specificare che questo non dà alcuna certezza sulla bontà dei nomi individuati e listati sul file di log, ovvero le identità rilevate coincidono *probabilmente* con gli attaccanti. I dati ottenuti sono indicativi e potrebbero essere utilizzati per l'esecuzione di ulteriori strategie di detection.

6.3.4.2 Beacon retardation

Descrizione L'attacco considerato in questa sezione consiste nell'apporto di un ritardo da parte degli attaccanti ai beacon in uscita dalla propria OBU che farà diventare di fatto inservibile l'informazione da essi trasportata. L'azione di disturbo dell'attaccante potrebbe inoltre generare una sorta di attacco DoS in quanto sia la rete che i nodi vittima saranno "invasi" da messaggi "inutili".

La funzione *doBeaconRetardationAttack()* è stata appositamente definita per simulare questa tipologia di attacco. Essa agendo in determinati istanti di tempo e per un certo intervallo di tempo, esegue una modifica dell'istante di trasmissione attraverso l'aggiunta di un ritardo (Δt) appositamente specificato e passato come parametro. L'utilizzo di una struttura dati come la HashMap risulta particolarmente utile ed efficace per far corrispondere ad ogni dato istante di tempo dell'attacco tutti i messaggi manipolati dall'attaccante in modo tale che al momento giusto, ovvero Δt istanti di tempo dopo la generazione e l'inserimento al suo interno, questi possano essere estratti e trasmessi alle vittime ignare.

Da quanto detto si può affermare che una strategia utile per la rilevazione di questa azione di disturbo può basarsi sulla verifica del contenuto dei messaggi in ingresso alla OBU ed in particolare del valore presente all'interno del campo *timestamp*. Poiché esso è stato oggetto di un "semplice" ritardo è sufficiente eseguire un controllo *locale* per valutarne l'*attualità* dei dati trasportati. Anche in questo caso un utente che si accorge di essere stato vittima dell'attacco aggiunge l'identità del nodo malizioso all'interno di una blacklist locale e ne rifiuta i messaggi futuri.

Risultati e considerazioni Il sistema che dorvebbe preoccuparsi della protezione della rete e degli utenti autorizzati della rete deve fare affidamento ad un componente sicuro ed affidabile, come un clock hardware, che in ogni istante possa fornirgli informazioni corrette relative al tempo affinchè esso possa effettuare con una certa sicurezza e accuratezza la verifica della coerenza dei dati presenti all'interno del campo timestamp nei pacchetti in ingresso. Nell'ambiente virtuale questo sistema viene implementato all'interno di una semplice funzione chiamata *tryToBlockBeaconRetardationAttack()* che si limita ad eseguire un controllo della data di creazione del pacchetto prima che questo possa essere "accettato" e salvato all'interno della tabella della base di dati che come noto simula la memoria delle OBU. L'informazione sul tempo viene prelevata dai dati delle tracce generate da SUMO dalle quali vengono di volta in volta prelevate le informazioni per la creazione dei messaggi. In questo modo tutti i beacon appositamente manipolati prima del loro inoltro sulla rete dagli attaccanti potranno essere rilevati evitando così di fornire al sistema di comunicazione un'informazione del tutto errata che potrebbe causare delle situazioni di pericolo in quanto i dati annunciati da questi messaggi si riferiscono a delle situazioni già avvenute in passato. In realtà, durante la fase di sviluppo, sono state definite e confrontate due diverse configurazioni della strategia appena descritta che in sostanza differivano soltanto per l'utilizzo di una blacklist locale sulla quale ogni nodo andava a salvare tutti i nodi smascherati. Entrambe le

soluzioni hanno dimostrato ottime capacità di protezione della rete ed hanno permesso di smascherare tutte le situazioni di attacco create anche se l'utilizzo della blacklist, nonostante permetta di bloccare completamente l'ingresso di dati provenienti da una controparte già etichettata come "inaffidabile", può generare un problema già esposto in un'altro contesto ovvero il rifiuto definitivo di un nodo autorizzato colpito anch'esso da un'azione di attacco che gli ha però sottratto l'identità.

I test finali sono stati condotti sulla strategia che fa uso di una blacklist locale. Questa individuando i dati manipolati ed inserendo le informazioni sugli attaccanti smacherati all'interno della lista permette al sistema di proteggersi in maniera *definitiva* dalle *avances* degli attaccanti noti anche negli istanti temporali successivi allo smascheramento in cui gli stessi attaccanti non stanno conducendo alcuna azione di attacco. La presenza della blacklist seppur richieda una certa quantità di memoria e comporti a lungo andare un decadimento delle prestazioni generali si rivela indispensabile poiché riesce a fornire sempre un importante aiuto al sistema di protezione che nella gran parte dei casi potrà quindi ridursi ad una semplice verifica dell'identità del nodo che ha generato il beacon.

Parametri richiesti dalla doBeaconRetardationAttack()

- Beacon da *manipolare*
- Identificativo di due attaccanti
- Durata dell'attacco per ognuno dei timestep specificati nella funzione
- Ritardo da "attribuire" al beacon da manipolare

Configurazione attacco e risultati delle simulazioni

- Attaccanti utilizzati per l'attacco: *Pertini_20_10* e *Audinot_7_0*
- Durata di ogni attacco: 5 secondi
- Ritardo: 2 secondi
- Durata della simulazione virtuale: 300 secondi
- Numero di test eseguiti: 10
- Indice di affidabilità (media) del sistema di protezione: 100% (grazie alla presenza della blacklist locale il sistema riesce sempre a bloccare più messaggi "fake" di quelli realmente creati)

6.3.4.3 Transmission range manipulation

Descrizione Questo attacco prevede la variazione dei valori relativi alla potenza del segnale e quindi del range di trasmissione definito dallo standard per poter consentire ad un utente malizioso di inviare i propri messaggi al di fuori della "normale area di pertinenza". Esso causa la ricezione di informazioni inappropriate per i nodi vittima in quanto i messaggi conterranno dei dati riferiti ad un *luogo* diverso da quello a cui essi potrebbero essere interessati.

La funzione *doTransmissionRangeManipulationAttack()* può essere utilizzata per compiere la simulazione di questo attacco. Questa riceve come parametri

gli identificativi di quattro attaccanti, la distanza attuale tra attaccante e vittima e un valore numerico che indica lo scostamento rispetto al range di trasmissione indicato inizialmente al lancio dell'applicazione SBX e grazie ad essi può generare e trasmettere virtualmente i messaggi a tutti coloro i quali risiedono all'interno della nuova "area di attacco".

Lo smascheramento di questa azione di disturbo potrà essere eseguito dalle vittime soltanto a seguito della verifica dei dati relativi alla posizione della sorgente all'interno dei beacon: se la distanza tra la vittima e l'attaccante è maggiore rispetto al range standard allora il messaggio relativo viene scartato.

Il blocco di codice che gestisce il *beaconing* all'interno della classe omonima effettua il logging dei pacchetti facendo affidamento al concetto di probabilità e di distanza, come spiegato nell'ultimo paragrafo del quinto capitolo. Poiché questo attacco rappresenta una forzatura al sistema e poiché tutti questi messaggi verrebbero di fatto bloccati non consentendo una corretta simulazione dell'attacco, si è preferito "concedere" un *pass* speciale ai pacchetti manipolati per permettergli di raggiungere "correttamente" le destinazioni designate.

Risultati e considerazioni La protezione della rete e degli utenti da un attacco di questo genere può avvenire mediante un sistema di rivelazione che consenta ad ogni nodo di verificare la validità dei dati trasportati da ogni beacon ricevuto. In particolare, come già anticipato, per smascherare questo attacco è sufficiente che ogni nodo valuti la distanza che intercorre tra sé stesso ed il sorgente del messaggio per deciderne il destino: se questa è compatibile con il range di trasmissione definito e specificato al lancio dell'applicazione SBX allora si procede col salvataggio in memoria, viceversa se ne effettua lo scarto. Questa semplice procedura è implementata dalla funzione *tryToBlockTransmissionRangeManipulationAttack()* presente all'interno della classe Attacks. Come in tutti gli attacchi fin qui considerati anche questo è stato testato a lungo al fine di valutare la validità del sistema di protezione sviluppato. Questo, seppur sia molto semplice, ha superato brillantemente ogni verifica permettendo di "catturare" la totalità dei messaggi manipolati creati e ottenendo di conseguenza un altissimo indice di affidabilità che ricordiamo essere definito dal rapporto tra il numero di pacchetti manipolati catturati ed il numero di pacchetti manipolati creati.

Parametri richiesti dalla doTransmissionRangeManipulationAttack()

- Identificativo di quattro attaccanti
- Identificativo della vittima
- Un valore Δ che va aggiunto al range di trasmissione predefinito

Configurazione attacco e risultati delle simulazioni

- Attaccanti utilizzati per l'attacco: *Pertini_20_10*, *Gandhi_40_0*, *XXI_Aprile_12_10* e *Gandhi_60_18*
- Δ=100 metri
- Durata della simulazione virtuale: 300 secondi
- Numero di test eseguiti: 10
- Indice di affidabilità (media) del sistema di protezione: 100%

6.3.5 Attacchi al ricevitore di beacon

6.3.5.1 Beacon suppression

Descrizione Quello della soppressione dei beacon è una tipologia di attacco già ampiamente considerata nella sezione precedente quando sono state discusse le vulnerabilità del trasmettitore di beacon e viene nuovamente riproposta perchè strettamente correlata ad un altro modulo della OBU suscettibile a questo tipo di attacco, ovvero il ricevitore di beacon. I beacon "gestiti" dal ricevitore di beacon rappresentano i dati che dovrebbero essere salvati all'interno dello storage della *scatola nera* e la loro soppressione totale o parziale ne causerebbe la decimazione. Un attaccante tramite la sua azione potrebbe voler sopprimere tutti i messaggi in ingresso in alcuni intervalli di tempo o parte di essi in accordo con una strategia precisa che potrebbe ad esempio voler "evitare" tutti i messaggi provenienti da una data sorgente.

La funzione che all'interno della classe Attacks si preoccupa di mettere in atto la simulazione di questo tipo di attacco prende il nome di *doSuppressionReceivedBeaconsAttack()*. Tra i parametri richiesti sono presenti gli identificativi degli attaccanti e delle vittime, un valore numerico che indica la durata dell'attacco a partire dai vari istanti di tempo prestabiliti e, per finire, un intero che definisce una delle "modalità di funzionamento" tra:

1. soppressione totale dei beacon in ingresso (modalità 1);
2. soppressione dei beacon provenienti da una data sorgente (modalità 2).

Questa strategia potrebbe essere messa in atto dagli attaccanti per evitare volutamente di salvare i messaggi per uno specifico scopo anche se un'analisi dei dati a posteriori potrebbe indurre a pensare che il ricevitore di beacon abbia potuto riscontrare un malfunzionamento durante la sua attività (ad esempio l'antenna di ricezione ha subìto un danneggiamento) o che il veicolo attaccante si sia trovato in determinati intervalli di tempo in una cosiddetta area sparsa. Questi sono i motivi principali per cui risulta difficile definire una tecnica efficace per lo smascheramento di questo attacco ai danni del sistema di comunicazione. Una possibilità potrebbe essere rappresentata dall'interpolazione dei dati trasmessi con quelli ricevuti: se negli stessi istanti temporali in cui un nodo A *dichiara* di non aver ricevuto alcun dato si dovesse scoprire che in realtà egli abbia generato e trasmesso dei beacon ai nodi "vicini" allora è semplice intuire che se vi è un problema sulla rete questo è rappresentato proprio dal nodo A. Purtroppo non sempre è possibile avere un accesso diretto sia ai dati trasmessi che a quelli ricevuti. Se prelevare i dati ricevuti è piuttosto semplice (basta accedere alla memoria interna alla propria OBU), accedere invece a quelli inviati non è così banale poiché significherebbe nella realtà dover accedere al database centrale o, caso peggiore, valutare il contenuto delle memorie di tutti i veicoli che negli istanti di tempo precedenti all'attacco si trovavano nell'intorno del veicolo attaccante per verificare che non vi siano dei messaggi loggati aventi come sorgente proprio il veicolo attaccante. Come è facile intuire nel mondo reale questa procedura non sarebbe di facile attuazione mentre nell'ambiente simulato, dove si ha la possibilità di accedere a

tutti i dati ricevuti/trasmessi da tutti i nodi buoni e non, risulta più facile da implementare.

Risultati e considerazioni Le considerazioni generali per un possibile sistema di rilevazione per questo tipo di attacco sono equivalenti a quelli già fatte per l'altro caso di Beacon Suppression riferito però al trasmettitore di beacon. In questo contesto il dispositivo interno alla OBU soggetto all'attacco è il ricevitore dei beacon e di conseguenza la situazione che si presenterà sulle memorie dei dispositivi sarà speculare a quella del caso precedente. Qui in particolare l'attaccante non vorrà non trasmettere beacon ai vicini ma semplicemente si rifiuterà di salvare in toto oppure in parte i dati provenienti dai vicini sulla propria memoria. Anche in questo caso un possibile sistema per la rilevazione dell'attacco di beacon suppression non può che essere un sistema globale che richieda l'accesso alla tabella della base di dati sulla quale sono memorizzati tutti beacon scambiati da tutti i nodi durante l'intero arco della simulazione e sia le strategie per l'analisi dei dati che le prestazioni e l'affidabilità rimangono gli stessi di quelli già riscontrati per l'altro caso di beacon suppression.

Parametri richiesti dalla doSuppressionReceivedBeaconsAttack()

- Beacon da *manipolare*
- Identificativo di due attaccanti
- Identificativo di due vittime
- Durata dell'attacco per ognuno dei timestep specificati nella funzione
- Modalità di funzionamento dell'attacco

Configurazione attacco e risultati delle simulazioni

- Attaccanti utilizzati per l'attacco: *Audinot_7_0* e *Costa_1_17*
- Vittime utilizzate per l'attacco: *Pepoli_3_0* e *Costa_12_0*
- Durata di ogni attacco: 30 secondi
- Durata della simulazione virtuale: 300 secondi
- Numero di test eseguiti: 10
- Il sistema di detection sviluppato permette di individuare correttamente i nodi scelti per l'esecuzione dell'attacco. Tuttavia è doveroso specificare che questo non dà alcuna certezza sulla bontà dei nomi individuati e listati sul file di log, ovvero le identità rilevate coincidono *probabilmente* con gli attaccanti. I dati ottenuti sono indicativi e potrebbero essere utilizzati per l'esecuzione di ulteriori strategie di detection.

6.3.5.2 Timestamping manipulation

Descrizione Il presente attacco, come dice il nome, consiste nella *falsificazione* dell'informazione relativa alla data di creazione dei beacon in ingresso al nodo attaccante e quindi nell'invalidazione dei dati originali presenti nei campi dei beacon inoltrati dai nodi autorizzati sulla rete. Un attaccante potrebbe decidere di eseguire una manipolazione della totalità dei beacon ricevuti o di parte

di essi al fine di inscenare, ad esempio, una truffa ai danni della compagnia assicurativa in seguito ad un incidente che lo ha visto coinvolto.

La funzione che simula quest'azione di disturbo è chiamata *doTimestamping-ManipulationAttack()* e, oltre al beacon da invalidare, riceve come parametri un valore che indica l'errore da apportare al dato presente all'interno del campo *timestamp* e due stringhe che rappresentano gli identificativi dei nodi che devono "eseguire" l'attacco.

Una strategia utile per l'individuazione di questo attacco potrebbe basarsi su un servizio che prima dell'operazione di storage dei messaggi sulla memoria della OBU effettui una verifica della validità dei dati contenuti all'interno dei vari campi del beacon ed in particolare della data di creazione e se questa dovesse non essere coerente con i valori di data e ora attuali (considerando anche l'effetto della latenza dei messaggi sulla rete) ne eviti la memorizzazione.

Risultati e considerazioni L'algoritmo da mettere in atto per la salvaguardia del sistema di comunicazione da un attacco di questo tipo assomiglia molto a quello già precedentemente implementato per il caso del Beacon Retardation e si basa sulla verifica della veridicità dei dati relativi alla data di generazione del messaggio posta all'interno del campo timestamp. Se l'attaccante tentasse di forzare il sistema modificando l'entità di questo valore si ha la possibilità, attraverso un semplice confronto, di evidenziare la criticità e decidere di non salvare quel beacon. Il sistema sviluppato si è rivelato molto funzionale ed efficace allo stesso tempo ed ha permesso nei test eseguiti di smascherare abilmente questo tipo di attacco rilevando i messaggi manipolati, evitandone così il salvataggio all'interno della memoria delle OBU.

Parametri richiesti dalla doSuppressionReceivedBeaconsAttack()

- Beacon da *manipolare*
- Identificativo di due attaccanti
- Durata dell'attacco per ognuno dei timestep specificati nella funzione
- Errore (in secondi) da "attribuire" alla data di creazione del beacon

Configurazione attacco e risultati delle simulazioni

- Attaccanti utilizzati per l'attacco: *XXI_Aprile_12_10* e *Gandhi_60_18*
- Durata di ogni attacco: 20 secondi
- Errore: 2 secondi
- Durata della simulazione virtuale: 300 secondi
- Numero di test eseguiti: 10
- Indice di affidabilità (media) del sistema di protezione: 100%

6.4 Composizione e configurazione Anomaly Checker

L'Anomaly Checker è un componente aggiuntivo al sistema di comunicazione standard rappresentato dalla On Board Unit che mette in atto delle strategie di controllo

per la cattura di eventuali anomalie sui dati in ingresso al ricevitore di beacon di ogni OBU permettendo così di creare il cosiddetto Anomaly Log che, nello specifico caso dell'applicazione sviluppata, corrisponderà ad una nuova tabella sulla base di dati che alla conclusione di ogni simulazione conterrà tutti i problemi rilevati su tutti dati ricevuti da ogni singolo nodo della rete mentre in un contesto reale corrisponderà ad una zona di memoria diversa da quella utilizzata dalla OBU per il salvataggio delle informazioni ricevute durante la fase di beaconing. La tabella appena citata è stata rinominata AnomalyLog e la sua struttura, piuttosto semplice, si compone di soli 5 campi:

- id dell'anomalia;
- id del nodo che ha generato l'anomalia;
- id del nodo che ha segnalato l'anomalia;
- tipologia di anomalia riscontrata;
- timestep in cui è avvenuta la rilevazione dell'anomalia.

Il principale punto di forza di questo ulteriore dispositivo all'interno del sistema di comunicazione è quello di mettere a disposizione un controllore o meglio una sorta di spia che non sia semplicemente capace di smascherare delle azioni di attacco in corso sulla rete ma che abbia soprattutto la possibilità e la capacità di comunicare in modo sicuro col mondo esterno ed in particolare con una terza parte fidata che in determinati istanti temporali possa essere avvisata per una qualche azione sospetta rilevata all'interno della rete affinché possa intervenire per "ammonire" il nodo malizioso, una volta individuato. Così come la On Board Unit, l'Anomaly Checker non avrà a disposizione un quantitativo di memoria illimitato sul quale poter salvare le informazioni e i dati e per tale motivo sarà indispensabile specificare un protocollo che indichi le modalità e i tempi con cui debbano essere eseguite queste operazioni di "scarico della memoria". Questa, nel caso dell'Anomaly Checker, potrebbe tuttavia avere un taglio diverso rispetto alla OBU in quanto si prevede che il volume delle anomalie riscontrate da un veicolo in un certo momento sulla rete non sia in alcun modo paragonabile a quello dei beacon ricevuti, soprattutto nel caso di rete densa. Idealmente il protocollo di comunicazione tra il dispositivo qui presentato e la controparte dovrebbe essere attuato ad intervalli di tempo molto più ristretti rispetto al caso della OBU trattandosi di dati piuttosto sensibili che potrebbero in qualche modo intaccare la sicurezza degli utenti durante i loro spostamenti. Questa idea implementativa nasce dalla seguente considerazione logica: se tutti i veicoli residenti in una certa area della rete dovessero individuare la stessa anomalia e riuscissero a comunicare tempestivamente quanto accaduto all'entità preposta al controllo allora quest'ultima potrebbe avviare in modo quasi istantaneo le giuste procedure nel tentativo di individuare il colpevole dell'azione sospetta. Se così fosse, quindi, la memoria richiesta da questo componente potrebbe essere irrisoria dal momento che essa si trasformerebbe più in una sorta di "buffer temporaneo" sul quale i dati dovrebbero risiedere per pochi e limitati intervalli temporali (nel caso ottimo in cui non vi siano problemi al collegamento di rete e in cui il veicolo riesca a trasmettere correttamente tutte le informazioni senza particolari ritardi).

Inizialmente si pensava che l'Anomaly Checker potesse limitarsi ad eseguire due semplici controlli con l'ausilio di altri moduli presenti sulla OBU:

- il controllo sulla data di creazione del beacon;
- il controllo sulla posizione geografica del nodo sorgente.

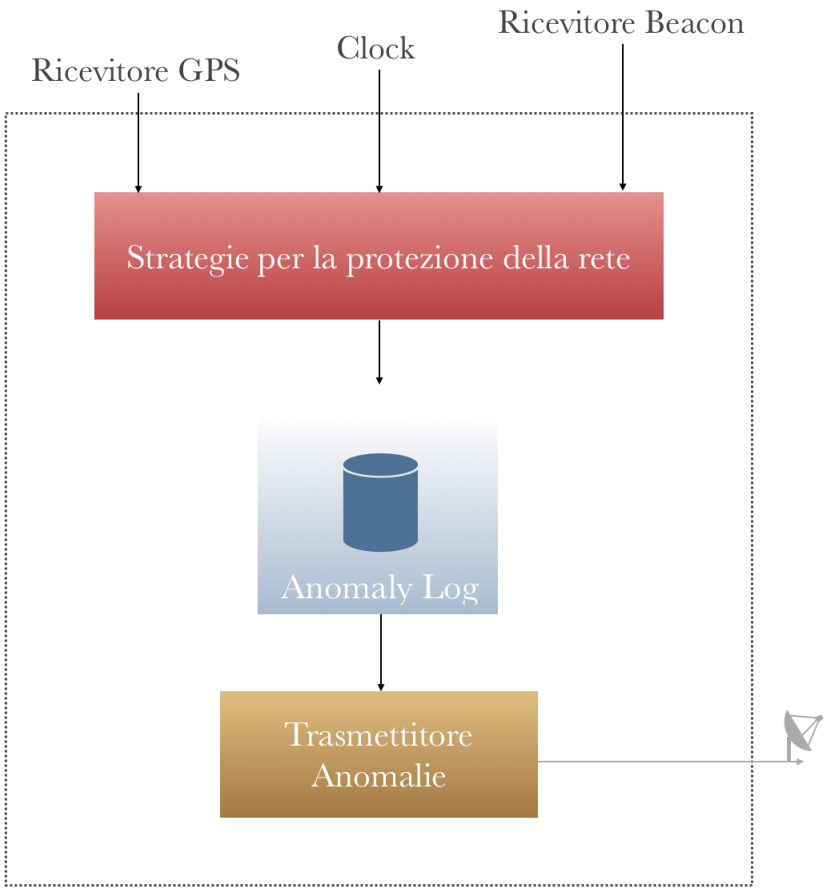


Figura 6.3: Composizione interna Anomaly Checker

I due controllori, che nella Figura 5.6 del precedente capitolo per una maggiore chiarezza erano stati divisi ed etichettati come *Controllore della posizione* e *Controllore del timestamp*, potevano avvalersi rispettivamente del segnale GPS captato dal ricevitore interno alla OBU e dell'informazione sul tempo fornita dallo stesso segnale GPS o da un clock eventualmente presente sulla OBU o sull'Anomaly Checker stesso. In seguito però, durante la fase di implementazione e sviluppo delle funzionalità di questi due elementi, ci si è accorti che le potenzialità di questo modulo *ideale*, che abbiamo deciso di introdurre all'interno del sistema di comunicazione standard, sono molte di più di quelle che avremmo potuto pensare nella fase di studio preliminare e per tale ragione si è deciso di trasformarlo in un mega-controllore all'interno del quale potessero essere racchiuse tutte le strategie individuate e descritte nel paragrafo precedente per la protezione della rete da ogni possibile attacco. Questa operazione permette di unire le potenzialità dei singoli schemi di protezione alla capacità di comunicazione (quasi) diretta con un'entità fidata, al momento anch'essa ideale, la

quale incrociando le segnalazioni dei singoli nodi nella varie zone della rete potrebbe individuare e ammonire i nodi che hanno causato dei danni alle comunicazioni. La descrizione appena fatta sull'Anomaly Checker trova un riscontro grafico nella Figura 6.3. Attraverso questa è possibile scorgere con maggiore chiarezza quali sono gli elementi costituitivi di questo importantissimo dispositivo e quali sono invece le linee di collegamento con i vari componenti interni alla On Board Unit richieste per un corretto svolgimento delle proprie operazioni. Rispetto alla configurazione iniziale è possibile notare come il blocco rinominato "Strategie per la protezione della rete" abbia preso il posto dei due controllori prima citati e, per finire, come l'aggiunta del "Trasmettitore delle anomalie", che era presente anche nella configuazione iniziale, completi lo schema funzionale dell'Anomaly Checker.

Le informazioni salvate sulla tabella AnomalyLog, che rappresenta il database delle anomalie globale nel sistema VANET, potrebbero rivelarsi indispensabili per uno scambio dati sicuro ed affidabile. Inoltre la presenza del dato temporale in corrispondenza di ogni anomalia rende possibile la definizione di una analisi ancora più approfondita e accurata che potrebbe aiutare gli enti preposti a scandagliare con una maggiore precisione il vicinato di ogni veicolo segnalato al fine di ricostruirne l'intorno e l'eventuale scenario d'attacco così da poter condurre una ricerca ancora più meticolosa anche attraverso il prelievo, lo studio e l'interpolazione dei dati prelevati dai beacon loggati dalle vittime degli attacchi individuati.

Alla stregua delle considerazioni appena fatte su questo importante componente verrebbe anche da pensare alla sicurezza dei dati da esso catturati e custoditi sul file di log. Esso infatti potrebbe rappresentare un obiettivo sensibile per gli attaccanti che, potendo in qualche modo accedervi, potrebbero volutamente manipolarne il contenuto attraverso l'aggiunta di nuove anomalie "fake" o la modifica di quelle già presenti affinché possano essere alterate le analisi condotte su di essi dalla controparte fidata. Un possibile esempio, seppur molto semplice, potrebbe essere rappresentato da uno scenario in cui un attaccante, sapendo di poter essere individuato dalle vittime durante l'azione di attacco, inserisce all'interno del log delle informazioni contrastanti appositamente studiate che possano creare dei disguidi e delle controversie durante l'esecuzione delle procedure di controllo delle anomalie. Se ad eseguire questa azione di attacco non fosse un solo attaccante ma un gruppo coordinato di daemon allora lo scenario diventerebbe più interessante e difficile da "studiare". Queste brevi valutazioni sono di assoluta importanza in quanto aiutano a capire quanto sia indispensabile securizzare questo dispositivo. Per certi versi la problematica appena affrontata può essere ricondotta a quella già trattata qualche capitolo addietro relativa alla protezione della memoria della OBU. Entrambe, dovendo "mantenere" al loro interno delle informazioni particolarmente sensibili, sono sicuramente degli ottimi obiettivi per quegli attaccanti che intendono "avvelenare" questi dati per trarne dei vantaggi personali o semplicemente per causare dei fastidi e dei danni alle comunicazioni interveicolari.

Ritornando quindi all'applicazione SBX ed in particolare alla sua composizione, l'Anomaly Checker corrisponderà ad una nuova classe i cui metodi si preoccupano semplicemente di implementare un protocollo di comunicazione con la base di dati. Questi verranno richiamati ogni qual volta un nodo, in seguito alla ricezione di un

beacon, dovesse individuare la presenza di qualche criticità che possa far presagire la presenza di un utente malizioso sulla rete. Trattandosi di semplici schemi di comunicazione opportunamente ottimizzati il carico apportato sarà minimo. All'interno della classe Beaconing verrà creato un oggetto AnomalyLog grazie al quale sarà possibile caricare le informazioni critiche rilevate sulla tabella della base di dati che alla fine della simulazione conterrà tutte le "lamentele" di tutti coloro i quali avranno individuato in ogni `timestep` un possibile attaccante.

6.5 Considerazioni finali

Il presente capitolo è stato quasi interamente dedicato allo studio e allo sviluppo delle simulazioni degli attacchi e dei relativi sistemi di rilevazione. La definizione di uno schema di difesa specifico per ogni tipo di attacco considerato ha permesso di valutare se fosse possibile o meno proteggere la rete VANET ed il sistema di comunicazione dalle azioni degli attaccanti. Partendo dalle considerazioni fatte nel capitolo precedente relativamente alle vulnerabilità di ogni singolo componente della On Board Unit, si è cercato di definire sia uno schema attraverso cui fosse possibile simulare virtualmente ogni scenario di attacco che una strategia di difesa che, attendendosi alle specifiche del problema, fosse capace di rilevarlo e possibilmente contrastarlo. In seguito alla valutazione dei risultati ottenuti dai numerosi test eseguiti si è potuto constatare come tutti i sistemi di protezione *locali* si siano dimostrati molto efficaci svolgendo un ottimo lavoro e restituendo costantemente degli indici di affidabilità superiori al 90% e in alcuni casi addirittura pari al 100%. Questo dato straordinario indica che, nei casi peggiori, ben 9 messaggi manipolati su 10 riescono ad essere individuati dagli schemi di difesa sviluppati. Per contro uno schema di difesa globale, il cui sviluppo si è rivelato necessario per una particolare tipologia di attacco, non può assicurare, come era prevedibile, una protezione totale agli utenti della rete. Questo infatti permette di individuare gli identificativi di quei nodi che durante la simulazione hanno avuto un comportamento giudicato per qualche motivo "strano" ma non per questo significa che essi siano dei veri attaccanti. Questa è la ragione principale per la quale non è possibile affermare che questi siano dei sistemi veramente affidabili. Tuttavia essi aiutano a valutare la qualità delle comunicazioni e ad individuare eventuali *nodi sospetti* e per questo motivo è giusto riconoscerne l'importanza. Le proprietà e le differenze tra i due metodi di studio degli attacchi sono ormai note e a questo punto, dopo aver valutato e discusso ogni possibile minaccia, diventa piuttosto facile procedere con una classificazione. A tal proposito nella Tabella 6.1 vengono riassunti e associati alla relativa classe di appartenenza (locale o globale) tutti gli algoritmi sviluppati per la rilevazione dei possibili attacchi al sistema di comunicazione considerati. Per finire, è doveroso spendere qualche parola anche per il cosiddetto "mega-controllore" che rappresenta indubbiamente un *must* per l'intero sistema di comunicazione. Questo infatti mettendo in atto la strategia di difesa definita per ogni tipo di attacco considerato su ogni beacon ricevuto potrebbe rilevare e di conseguenza comunicare la presenza di qualche azione sospetta sulla rete ad una terza parte fidata che ricevendo queste segnalazioni dovrebbe avviare tutte le verifiche del caso affinchè sia possibile identificare quel nodo o quei nodi che si pensa siano stati la causa della strana situazione

rilevata e procedere eventualmente con la loro espulsione dalla rete.

Attacco	**Tipologia sistema di protezione**
GPS Spoofing	Locale
GPS Blocking	Locale
Fake Data	Locale
Active Flooding	Locale
Beacon (sended) Suppression	Globale
Beacon Retardation	Locale
Transmission Range Manipulation	Locale
Beacon (received) Suppression	Globale
Timestamping Manipulation	Locale

Tabella 6.1: Classificazione dei sistemi di protezione relativi ad ogni attacco

Capitolo 7

Conclusioni

Lo studio della tematica trattata nella presente tesi nasce dall'interesse per il settore nel quale si colloca e dall'importanza innata di questa tipologia di rete che nel futuro prossimo potrebbe ricoprire un ruolo fondamentale nella salvaguardia dell'incolumità degli utenti sulle strade di tutto il mondo.

Numerosi sono stati gli studi e le ricerche effettuate in questo campo nel corso degli anni per valutare e quindi ottimizzare ogni singolo aspetto e ogni singola criticità di una Vehicular Ad Hoc Network. Essa, essendo principalmente stata pensata per *eseguire* delle applicazioni di sicurezza, richiede una particolare attenzione sia relativamente alla qualità del canale di comunicazione sul quale verranno trasmessi i beacon e sia relativamente alla protezione degli utenti e dei dati ad essi associati e da essi inoltrati sulla rete. Poiché la banda a disposizione non è illimitata e poiché la propagazione del segnale nell'aria può facilmente essere soggetta a degrado a causa di collisioni o interferenze, si è ritenuto indispensabile procedere con la definizione e lo sviluppo di schemi appositi per prevenire la perdita dei pacchetti e quindi migliorare la trasmissione dei dati cosicché possano essere evitati, o almeno limitati, problemi piuttosto gravi che potrebbero addirittura rappresentare la causa di gravi incidenti stradali e quindi di vittime. Ogni messaggio di sicurezza può registrare una latenza di trasmissione pari a qualche decimo di secondo. Questo implica che un beacon, data l'importanza dei dati trasportati, non può venire bloccato o ritardato per qualche strana ragione in quanto potrebbe diventare inutile ai fini della sicurezza degli utenti e le informazioni potrebbero risultare "scadute" una volta consegnate ai destinatari. Affinché questo requisito possa essere rispettato si è ritenuto indispensabile procedere con la definizione di strategie ad hoc che potessero ottimizzare il flusso dei pacchetti generati dalle applicazioni di sicurezza a favore ad esempio di quelli appartenenti alle applicazioni di infotainment, che indubbiamente ricoprono un'importanza minima rispetto alle prime. Per far ciò gli ideatori dello standard WAVE hanno pensato di suddividere lo spettro di frequenze a disposizione in tanti slot o canali, ognuno dei quali potesse essere "affidato" in modo specifico ad una piuttosto che all'altra tipologia di applicazione onde evitare situazioni di sovrapposizione che potessero causare dei disagi al processo di beaconing.

Tra gli aspetti da considerare durante lo studio di fattibilità e sviluppo di una rete veicolare spiccano con prepotenza la privacy degli utenti e la sicurezza delle

comunicazioni. La protezione dei singoli utenti e dei loro dati da utenti malintenzionati è considerata una delle problematiche più ardue e complicate da affrontare e risolvere. Un attaccante potrebbe minacciare la "serietà" della rete VANET sfruttandone le debolezze. Un gran numero di ricerche sono state condotte in merito a queste tematiche nella speranza di poter giungere alla definzione di strategie ottime che riuscissero a limitare le vulnerabilità proprie di una rete di questo tipo. Dal momento che un attaccante fingendosi un nodo buono potrebbe decidere di bloccare la trasmissione di beacon o di iniettare sulla rete dei beacon volutamente manipolati diventa importante, se non indispensabile, condurre uno studio approfondito su quelle che potrebbero essere le modalità di attacco e le conseguenze per gli utenti della rete unitamente alle possibili strategie da mettere in atto per una protezione totale al fine di evitare, ad esempio, la creazione di scenari di traffico fittizi che potrebbero indurre gli utenti ad eseguire delle manovre scellerate che a loro volta potrebbero diventare la causa di incidenti. Proprio queste sono state le considerazioni che ci hanno incoraggiato a procedere allo studio e alla valutazione delle vulnerabilità che ogni malintezionato potrebbe sfruttare per minacciare la rete e limitarne le funzionalità.

Dopo una analisi iniziale molto ricca ed esaustiva sulle proprietà e sulle caratteristiche di una rete VANET utile per introdurre al meglio il lettore nel mondo delle reti veicolari, ci si è soffermati a lungo sugli aspetti legati alla sicurezza, a partire dalla tutela della privacy degli utenti e dagli standard di comunicazione ad oggi presenti. Come si è potuto notare lo stato attuale della ricerca in questo specifico campo è in continuo fermento e nuove soluzioni vengono continuamente sviluppate e presentate dai numerosi gruppi di ricerca presenti in tutto il mondo interessati a questa particolare tematica. Purtroppo però è doveroso affermare che gli sforzi fin qui fatti non sono stati sufficienti per procedere con la realizzazione e la messa in pratica in larga scala di questa importantissima tecnologia.

Ritornando al lavoro svolto e qui presentato, tutto è partito dalla simulazione di uno scenario di traffico cittadino mediamente popolato e trafficato sul quale sono stati appositamente inseriti dei veicoli attaccanti e dal popolamento della tabella della base di dati implementata con tutte le tracce e i beacon generati e scambiati virtualmente dai nodi che han preso parte alla simulazione grazie all'aiuto dell'applicazione SimulBeaconXchange sviluppata. A seguire poi molto spazio è stato dato alla presentazione delle diverse tipologie di attacco al sistema di comunicazione, alla loro simulazione e mitigazione. L'applicazione SBX è stata quindi ulteriormente modificata (nella sua fase iniziale la sua unica sola peculiarità era l'esecuzione del processo di beaconing) per poter definire un ambiente virtuale sul quale fosse possibile studiare le varie modalità di attacco attraverso delle simulazioni ad hoc. In seguito allo studio dei singoli attacchi ed alla realizzazione dei relativi schemi di protezione della rete, si è deciso di procedere con una classificazione dei sistemi di rilevazione. Questa ha permesso di mettere in risalto le capacità della prima classe, rappresentata dai sistemi di rilevazione locali, nel poter individuare e quindi contrastare un attacco mentre questo è in corso, e della seconda classe, quella dei sistemi di rilevazione globali, nel poter individuare i possibili colpevoli delle azioni di attacco riscontrate solo mediante un processamento a posteriori dei dati presenti sui beacon scambiati durante la simulazione. L'obiettivo finale, in questo caso, era quello di studiare meticolosamente ogni possibile "schema di attacco" affinché si potesse

ottenere un sistema di protezione completo che consentisse alla rete e agli utenti di difendersi da ognuno dei singoli attacchi considerati. Questo studio ha ritenuto indispensabile disporre di un ulteriore dispositivo all'interno del sistema di comunicazione che avesse la capacità, a seguito di alcune verifiche sui beacon prelevati dal ricevitore di beacon dettate dal sistema di rilevazione dell'attacco considerato, di "sollevare" delle anomalie che potessero poi essere trasmesse ad una controparte fidata per la sua valutazione attraverso l'interpolazione con tutte le altre anomalie eventualmente individuate da altri nodi.

Dai test eseguiti e dai risultati ottenuti, già ampiamente valutati nel capitolo precedente, è stato possibile ottenere un riscontro sulla qualità e sulla validità di ogni singola strategia di protezione implementata. Nel complesso ritengo che i risultati ottenuti siano soddisfacenti o addirittura ottimi in certi casi. Sebbene le strategie per la mitigazione di un attacco di beacon suppression non si siano dimostrati efficaci al 100% offrendo "soltanto" la lista dei probabili attaccanti, le restanti hanno sempre dimostrato un'ottima capacità nel bloccare localmente l'attacco sferrato riducendo in modo sostanziale il rischio di salvataggio e accettazione di messaggi manipolati da parte delle vittime inconsapevoli.

Il sistema di logging, rappresentato dalla OBU, è stato ampiamente valutato e considerato in ogni suo aspetto tranne che per una particolare caratteristica costruttiva: la "capacità della memoria" per lo storage temporaneo dei beacon. Anni addietro, quando è stato avviato il percorso di studio per lo sviluppo della VANET, uno dei limiti con cui dover combattere era proprio quella della memoria a disposizione delle OBU e per tale motivo numerosi *algoritmi di scarto dei pacchetti* sono stati sviluppati. Questi, sulla base di metriche specifiche che si affidavano ad esempio al concetto di *vicinanza dei nodi*, decidevano quali pacchetti dovevano e potevano essere "accettati" in ogni timestep da ogni OBU affinchè non venisse compromessa l'attività di altre applicazioni che, agendo sulle informazioni loggate da ogni nodo e interpolandole opportunamente, una volta giunte al database centrale, potessero ricostruire l'intorno di ogni veicolo in ogni istante di attività al fine di studiare, ad esempio, un eventuale scenario di attacco o di *spostamento sospetto* di qualcuno dei veicoli presenti. I *casi di studio* di quegli anni forzavano e limitavano fortemente la quantità di pacchetti che ogni veicolo potesse loggare in un'ora di attività definendo un tetto massimo di circa 40/50 MB che si traduceva in un utilizzo di memoria poco superiore a 1GB nel caso di una giornata intera di *spostamento* del veicolo. Nel corso degli anni però il costo della memoria è andato via via diminuendo e, ad oggi, questo potrebbe non essere più considerato un vero e proprio problema, anche se l'obiettivo di mantenere quanto più bassi possibili i costi di produzione di questi dispositivi continua ad essere una sfida attuale. Naturalmente la presente tesi esula dalle considerazioni relative al problema dei costi di produzione delle OBU in quanto tale problematica non risulta essere del tutto affine con l'ambito di sviluppo qui considerato. Questi infatti, come tutti i prodotti in commercio, quando avranno un largo impiego registreranno un andamento dei prezzi che varierà in base a tante possibili situazioni di mercato che potranno essere influenzate da un gran numero di fattori diversi, come la concorrenza tra i costruttori ad esempio. Considerato che la memoria dovrebbe venire svuotata ogni qual volta il veicolo si trova in sosta e considerato che solo in casi estremi un veicolo si ritroverà in continuo movimento per più di un'intera giornata, oggi può essere più che accettabile disporre sulla OBU di

una memoria di 4 o addirittura 8 GB che certamente non risulta essere così costosa come qualche anno addietro.

Il progetto sviluppato può essere considerato un punto di partenza per la definizione di nuove applicazioni. Questo potrebbe essere ulteriormente esteso con la creazione e l'aggiunta di nuove tecniche e di nuovi schemi di difesa per la protezione della rete da possibili attacchi ed attaccanti. Un ulteriore passo in avanti potrebbe ad esempio riguardare la valutazione della validità e dell'efficacia delle strategie di rilevazione degli attacchi in un ambiente ostile dove il tasso di perdita di pacchetti, essendo molto più elevato, potrebbe influenzarne il comportamento che eventualmente potrebbe essere ridefinito e rivisto proprio per consentire una corretta esecuzione in ogni tipologia di ambiente.

Grazie alle informazioni relative agli spostamenti dei veicoli durante l'intero arco della simulazione presenti sulla base di dati sarà possibile sviluppare nuove soluzioni che possono, ad esempio, studiare e valutare la presenza di incidenti o di strane situazioni avvenute in seguito alla manipolazione dei dati poiché non bisogna dimenticare che non tutti i sistemi di rilevazione riescono a combattere a caldo un attacco in corso sulla rete e soprattutto non tutti i sistemi riescono a bloccare la totalità dei beacon manipolati generati dagli attaccanti. Questi ultimi inoltre, che non hanno ottenuto un indice di affidabilità massimo, potrebbero a loro volta esser messi sotto la lente di ingrandimento nel tentativo di ottenerne altri ancora più efficaci ed affidabili.

Giunti ai titoli di coda dell'intera trattazione vorrei poter esprimere le mie impressioni personali sulla tematica studiata dalla presente tesi. Nel corso degli ultimi 2-3 anni l'industria automobilistica è riuscita, grazie all'aiuto di specialisti appartenenti alle varie branche dell'ingegneria, a definire un'ampia varietà di sistemi di sicurezza attiva alla guida. Questi aiutano e assistono il conducente di un veicolo durante gli spostamenti riuscendo, ad esempio, a segnalare la presenza di un ostacolo o di un pedone sulla strada fino a fermarne addirittura la corsa onde evitare la collisione oppure a mantenere la corsia percorsa dall'automobile stessa che potrebbe ad esempio venire "abbandonata" in seguito ad un colpo di sonno o ad un malore del conducente. Questi appena fatti sono solo un paio di esempi e volendo se ne potrebbero fare tanti altri. Tutti questi sistemi, che erano inizialmente presenti soltanto sulle auto premium, stanno diventando ormai di serie anche sulle auto più piccole in quanto aiutano ad evitare situazioni talvolta catastrofiche. Essi infatti riescono effettivamente a salvaguardare l'incolumità dei conducenti e degli altri utenti della strada. Questo è il motivo che mi spinge a pensare all'importanza e all'utilità che potrebbe ricorpire una rete veicolare come quella studiata nel presente documento. Inoltre, visto che il futuro vedrà sicuramente la crescita e lo sviluppo della cosiddetta IoT e visto che numerosi progetti di ricerca sono finalizzati proprio allo studio dell'applicazione dell'IoT nell'ambito cittadino permettendo così la creazione di quella che viene spesso definita *smart city*, penso che ci possa essere anche una cooperazione tra le due tecnologie al fine di securizzare gli spostamenti degli utenti che potrebbero ad esempio essere avvisati di un semaforo che sta per diventare rosso o di un'improvviso malfunzionamento all'impianto semaforico in corrispondenza di un incrocio che potrebbe sicuramente esser la causa di incidenti fatali. La mia idea è che l'implementazione di una rete VANET potrebbe veramente cambiare il modo

di guidare ed il modo di concepire l'automobile che, così facendo, diventerebbe un oggetto ancora più smart di quanto non lo sia già. Se ogni utente riuscisse a sapere con largo anticipo ciò che sta accadendo a 100/200 metri di distanza da esso, un gran numero di incidenti potrebbe essere sicuramente evitato. Inoltre tutta una serie di applicazioni per la sicurezza attiva potrebbero essere attivate e rese disponibili agli utilizzatori di questa rete. Purtroppo però ancora oggi tanti sono i punti oscuri che necessitano di ulteriore tempo e sviluppo e per tale ragione credo che bisognerà attendere ancora un pò di anni prima che questa tecnologia possa finalmente vedere la luce.

Appendice A

Manuale del programmatore

L'implementazione della soluzione proposta nella presente tesi per la simulazione delle funzionalità proprie di una rete VANET consta in uno script per l'automatizzazione dell'intero processo e in una applicazione Java chiamata *SimulBeaconXchange*.

A.1 Script per l'esecuzione del processo

Il presente script è stato realizzato nell'intento di mettere a disposizione dell'utente finale e del programmatore un unico elemento attraverso il quale fosse possibile avviare in maniera totalmente automatica l'intero processo che simula le proprietà di una rete VANET ed effettuare il tuning di alcuni parametri utili per una corretta esecuzione. In linee generali esso si preoccupa di generare le tracce tramite il noto simulatore di traffico SUMO, creare e simulare lo scambio di beacon tra tutti i nodi della rete a partire dalle suddette tracce, eseguire l'attacco prescelto e la relativa strategia di protezione sviluppata e, per finire, salvare tutte queste informazioni su una base di dati. Lo script di cui si sta parlando è stato rinominato *RunProcess.bash* ed è rappresentato di seguito.

```
#!/bin/sh

MY_PATH=`` `dirname \``$0\``` ``

_db=``tesi_db''
_db_user=``root''
_db_password=``password''
_thread_num=``4''
_range=``300''
_attack_code=``code''
_simulation_duration=``duration''
```

```
function generateFakeTrace() {

sumo -c $MY_PATH/runfake.sumo.cfg --fcd-output $MY_PATH/fake.xml
echo XML file with FAKE traces created

/opt/local/bin/sumo-xml2csv $MY_PATH/fake.xml
echo FAKETrace file converted

/usr/local/mysql/bin/mysql -u $_db_user -p $_db_password $_db
--local_infile=1 -e ‘‘LOAD DATA LOCAL INFILE ‘‘‘ $MY_PATH/fake.csv’’’
INTO TABLE traces FIELDS TERMINATED BY ‘;’ LINES TERMINATED BY
‘\n’ IGNORE 2 LINES’’
echo Uploaded all fake traces into DB

/usr/local/mysql/bin/mysql -u $_db_user -p $_db_password $_db -e
‘‘DELETE FROM traces WHERE vehicle_id=‘‘ ’’
echo Null traces deleted

/usr/local/mysql/bin/mysql -u $_db_user -p $_db_password $_db -e
‘‘UPDATE traces SET fake_trace = ‘1’ ’’
echo Fake_Trace attribute setted on traces table

}

function generateRealTrace() {

sumo -c $MY_PATH/runreal.sumo.cfg --fcd-output $MY_PATH/real.xml
echo Simulation ended and XML file created

/opt/local/bin/sumo-xml2csv $MY_PATH/real.xml
echo Trace file converted

/usr/local/mysql/bin/mysql -u $_db_user -p $_db_password $_db
--local_infile=1 -e ‘‘LOAD DATA LOCAL INFILE ‘‘‘$MY_PATH/real.csv’’’
INTO TABLE traces FIELDS TERMINATED BY ‘;’ LINES TERMINATED BY ‘\n’
IGNORE 2 LINES’’
echo Uploaded all traces into DB

/usr/local/mysql/bin/mysql -u $_db_user -p $_db_password $_db -e
‘‘DELETE FROM traces WHERE vehicle_id=‘ ’ ’’
echo Null traces deleted

}
```

```
function callSBX() {

   java -jar $MY_PATH/SBX.jar $1 $2 $3 $4 $5 $6 $7
   echo Beacons uploaded into DB

}

function cleanTraceTable() {

   /usr/local/mysql/bin/mysql -u $_db_user -p $_db_password $_db -e
   ‘‘TRUNCATE traces’’
   echo Trace table cleaned

}

function deleteFiles() {

   rm -Rf $MY_PATH/real.xml
   rm -Rf $MY_PATH/real.csv
   rm -Rf $MY_PATH/fake.xml
   rm -Rf $MY_PATH/fake.csv
   rm -Rf $MY_PATH/e1_output.xml
   rm -Rf $MY_PATH/sumo_log.txt
   rm -Rf $MY_PATH/tripinfos.xml

}

#START PROGRAM
echo START!
echo ‘‘Do you want to create new traces with SUMO?’’
echo ‘‘Please type [y] or [n]’’

read response

case $response in
   ‘‘y’’)
      cleanTraceTable
      generateRealTrace &
      generateFakeTrace &
      wait
      deleteFiles
      callSBX $_db_user $_db_password $_db $_thread_num $_range
               $_attack_code $_simulation_duration;;
```

```
    ‘‘n’’)
       callSBX $_db_user $_db_password $_db $_thread_num $_range
               $_attack_code $_simulation_duration;;

    *)
       echo ‘‘Wrong digit.  Process aborted!’’;;

esac
echo FINISH!
exit
```

Le righe iniziali dello script appena mostrato contengono le definizioni di una serie di variabili che possono essere settate a piacimento dallo sviluppatore. In particolare le variabili sono:

- `_db` contiene il nome della base di dati che si vuol utilizzare;
- `_db_user` e `_db_password` rappresentano le credenziali di accesso alla base di dati utilizzata;
- `_thread_num` racchiude un valore numerico indicante il massimo numero di thread che possono essere eseguiti in parallelo dall'applicazione SBX per migliorarne le prestazioni;
- `_range` rappresenta il range di trasmissione utilizzato dall'applicazione SBX per lo scambio dati tra veicoli;
- `_attack_code` individua la tipologia di attacco che si vuol simulare;
- `_simulation_duration` indica la durata della simulazione.

Tutte queste informazioni appena presentate, ad esclusione del nome della base di dati, sono di assoluto rilievo in quanto rappresentano i dati in input richiesti dall'applicazione SBX per un corretto svolgimento delle funzionalità implementate al suo interno.

Oltre alle definizioni delle variabili, lo script comprende la definizione di quattro diverse funzioni che svolgono altrettanti compiti:

- ***generateRealTrace()***
  Questa funzione si preoccupa in sostanza di richiamare SUMO per generare le tracce reali, ovvero quelle tracce che identificano gli utenti *buoni* all'interno della rete, tradurre il file XML creato in precedenza nel formato CSV attraverso il tool xml2csv e, per finire, caricare queste stesse informazioni sulla base di dati eliminando eventuali righe contenenti dati “non validi”. L'esecuzione della simulazione in SUMO richiede un file di configurazione (runreal.sumo.cfg in questo caso) che definisce le “condizioni” della simulazione stessa sullo scenario preimpostato della Città di Bologna che è stato oppotunamente riconfigurato per poter consentire in questo specifico caso di dar vita alle sole tracce dei veicoli “buoni” all'interno della rete.

- ***generateFakeTrace()***
  La presente funzione svolge gli stessi compiti della *generateRealTrace()* con la differenza che le tracce qui create identificano gli utenti *maliziosi* che si muovono all'interno dello stesso scenario cittadino. Queste particolari tracce saranno etichettate con un valore specifico nel campo "fake_trace" della tabella *Traces* sulla base di dati utilizzata al fine di consentirne l'identificazione nel corso delle elaborazioni successive. Anche in questo caso la simulazione con SUMO viene lanciata mediante l'uso di un nuovo file di configurazione appositamente creato, rinominato runfake.sumo.cfg, il quale permette di impostare l'ambiante virtuale che alla fine consentirà di ottenere in uscita i dati riferiti alle tracce dei soli veicoli attaccanti all'interno della rete.

- ***deleteFiles()***
  Questa funzione è stata creata al fine di evitare che file di grosse dimensioni generati nei vari step del processo e salvati all'interno della directory nella quale risiede lo script, possano restare inutilmente sulla memoria del sistema anche dopo l'elaborazione ed il caricamento delle informazioni ivi contenute sulla base di dati.

- ***callSBX()***
  La chiamata di questa funzione consente allo script di avviare automaticamente l'applicazione *SimulBeaconXchange* per simulare lo scambio di beacon tra tutti i nodi della rete a partire dalle tracce già salvate in precedenza sulla base di dati e i vari attacchi già ampiamente trattati nel Capitolo 6 del presente documento.

- ***cleanTraceTable()***
  Questa funzione consente allo script di ripulire in modo automatico la tabella della base di dati che contiene le tracce generate da SUMO in una precedente esecuzione della simulazione.

Le funzioni appena descritte, se così possono essere chiamate, permettono un'ottima organizzazione del codice all'interno dello script. Queste vengono chiamate una alla volta per poter svolgere il proprio compito. L'utilizzo dell'operatore "*&*", nel caso della *generateRealTrace()* e della *generateFakeTrace()*, permette una parallelizzazione dei due flussi di lavoro che, così facendo, saranno eseguiti contemporaneamente ottenendo un miglioramento delle prestazioni.

La base di dati usata in questo contesto, come già detto nel Paragrafo 5.2 del Capitolo 5, è una delle più importanti e diffuse a livello mondiale, ovvero MySQL. Grazie alla sua affidabilità e facilità di configurazione e utilizzo si ha la possibilità di poter sviluppare le proprie applicazioni senza particolari difficoltà e con ottime prestazioni.

L'utilizzo di questa soluzione nel presente progetto richiede quindi la presenza sul sistema di MySQL. Per semplicità verrà fornito un file con estensione *.sql*, chiamato tesi_db.sql, che descrive la struttura interna delle tabelle del database. In questo modo basterà importare questo file di configurazione dal pannello di controllo di MySQL e modificare opportunamente le variabili all'interno dello script, se fosse il

caso, per poter replicare esattamente il progetto qui presentato sul proprio sistema. La struttura interna delle tabelle della base di dati è stata già esposta e discussa ampiamente nel Capitolo 5 anche se nel paragrafo successivo per completezza sarà mostrato il codice che consente di replicarla efficacemente senza l'importazione di alcun file. In particolare la base di dati, chiamata nello specifico caso *tesi_db*, contiene quattre tabelle, rinominate *Traces*, *Beacons*, *Correspondance* e *AnomalyLog* che contengono le informazioni elaborate ai vari step dell'applicazione qui sviluppata e presentata.

A.2 Applicazione *SimulBeaconXchange*

L'applicazione SBX è stata creata con l'intento di poter definire un ambiente virtuale personalizzato che non facesse uso di particolari tool esterni pronti all'uso e che consentisse di simulare le attività proprie di una Vehicular Ad Hoc NETwork. Affinché questa possa assolvere alle funzionalità definite al suo interno mediante le diverse funzioni e le diverse classi è indispensabile che vengano forniti al lancio ben 5 parametri i quali, in ordine di inserimento, riguardano lo username e la password per accedere alla base di dati (il cui nome può essere settato direttamente all'interno della funzione dbConnect() definita nella classe Functionality), il numero di thread che si vogliono eseguire in parallelo, il valore del range di trasmissione e, per finire, il codice dell'attacco che si vuol simulare sulla rete. Inizialmente essa svolgeva una sola funzione, ovvero simulava il processo di beaconing a partire dalle tracce generate da SUMO, e successivamente è stata ampliata implementando le strategie per le simulazioni delle diverse tipologie di attacco che possono essere messe in atto dagli attaccanti per infrangere le funzionalità della rete. Tra le classi sviluppate è possibile trovarne due in particolare che rispettivamente definiscono l'oggetto *Beacon* e l'oggetto *Traccia*. Le proprietà dei due oggetti, già valutate e mostrate nel Capitolo 5, corrispondono in parte ai campi della tabella della base di dati che li ospita e sono sommariamente raccolte nelle tabelle a seguire (Tabella A.1 e Tabella A.2). Un programmatore, volendo, potrebbe espandere ancora le due classi e quindi le proprietà dei due oggetti per aggiungerne di nuove adatte al proprio scopo. Inoltre è presente un'altra classe ancora chiamata Functionality contenente gli strumenti utili per svolgere, come dice il nome stesso, alcune importanti funzioni utili nei vari punti del programma come il calcolo della distanza tra due nodi o il prelievo delle tracce dal database. Anche questa classe potrebbe essere ulteriormente ampliata attraverso l'aggiunta di nuovi metodi che possono ad esempio permettere l'esecuzione di nuove funzionalità all'interno della rete. Oltre alle classi appena indicate sono presenti anche la classe AnomalyLog e la classe Correspondance. La prima implementa le funzionalità dell'AnomalyChecker mentre la seconda modella la relazione nodo sorgente-nodo destinatario.

Il Main rappresenta il corpo principale dell'intera applicazione ed ha il compito di coordinare tutte le varie componenti al fine di poter consentire un corretto svolgimento delle funzioni proprie dell'applicazione stessa. Esso coordina l'esecuzione dei vari moduli e utilizza un ExecutorService per la gestione ottimale dei thread al fine di poter sfruttare al meglio le funzionalità definite all'interno della classe Beaconing che rappresenta il vero fulcro dell'applicazione. Essa, estraendo tutte le

tracce riferite ad un certo timestep, valuta per ogni veicolo prelevato il suo vicinato e ne definisce i beacon i quali partendo dallo stesso giungono ai destinatari che risiedono all'interno dell'area indicata dal range di trasmissione stabilito dall'utente al lancio dell'applicazione. Qui il programmatore potrebbe voler modificare l'area stratificata descritta nell'ultima parte del quinto capitolo che implementa il sistema di logging dei pacchetti basato sulla probabilità di ricezione che varia in relazione della distanza che intercorre tra i nodi che intendono comunicare. In particolare nella mia configurazione ho scelto di generare 10 sezioni circolari ognuna avente una data "probabilità di ricezione" associata mentre qualcun altro potrebbe decidere di variare questi parametri per cambiare il comportamento del sistema di comunicazione sviluppato. Inoltre questa importantissima classe permette di richiamare ed eseguire una particolare tipologia di attacco che una volta simulato per l'intera durata della simulazione restituirà un resoconto all'interno di un file di log contenente il risultato dell'operazione eseguita. La scelta dell'attacco da simulare viene gestita mediante uno switch che decide in base al codice specificato al lancio dell'applicazione (da 0 fino a 13) quale metodo richiamare (un sommario con i codici dei vari attacchi è presente nella Tabella A.5). L'implementazione degli attacchi vera e propria è invece definita all'interno di una classe ad hoc chiamata Attacks. Al suo interno sono presenti sia le strategie di attacco esposte nel Capitolo 6 che quelle utili per la protezione della rete da ognuno di questi. Se si vuol modificare il *comportamento* di un attacco o di un sistema di rilevazione in particolare oppure se si vuol aggiungere qualche altra tipologia di attacco a quelli già presenti è sufficiente agire sulla classe Attacks e modificarla secondo i propri bisogni.

A.3 Struttura e composizione database

La parte conclusiva di questa importante parte del documento è dedicata alla base di dati utilizzata dall'applicazione per lo svolgimento dei propri compiti. L'organizzazione delle tabelle è molto semplice e, volendo fornire un aiuto al programmatore, viene di seguito mostrato il codice MySQL che consente facilmente di ricrearle e replicarle. Queste, ovviamente, potranno essere liberamente modificate secondo le varie esigenze con l'aggiunta o la cancellazione di uno o più campi. Il codice appena citato è il seguente:

```
CREATE TABLE ‘beacons’ (
   ‘beacon_id’ int(10) NOT NULL,
   ‘id_src’ varchar(100) NOT NULL,
   ‘id_dst’ varchar(100) NOT NULL,
   ‘payload’ varchar(100) NOT NULL,
   ‘timestamp’ double NOT NULL,
   ‘trace_id’ int(10) NOT NULL,
   ‘distance’ double NOT NULL,
   ‘src_pos_x’ double NOT NULL,
   ‘src_pos_y’ double NOT NULL,
   ‘src_speed’ double NOT NULL )
```

```
CREATE TABLE ‘traces’ (
   ‘timestep’ double NOT NULL,
   ‘vehicle_angle’ double NOT NULL,
   ‘vehicle_id’ varchar(100) NOT NULL,
   ‘vehicle_lane’ varchar(100) NOT NULL,
   ‘vehicle_pos’ varchar(100) NOT NULL,
   ‘vehicle_slope’ double NOT NULL,
   ‘vehicle_speed’ double NOT NULL,
   ‘vehicle_type’ varchar(100) NOT NULL,
   ‘vehicle_x’ double NOT NULL,
   ‘vehicle_y’ double NOT NULL,
   ‘fake_trace’ int(1) NOT NULL DEFAULT ‘0’,
   ‘trace_id’ int(10) NOT NULL )

CREATE TABLE ‘correspondance’ (
   ‘id_correspondance’ int(10) NOT NULL,
   ‘id_vehicle_1’ varchar(100) NOT NULL,
   ‘id_vehicle_2’ varchar(100) NOT NULL )

CREATE TABLE ‘anomaly’ (
   ‘anomaly_id’ int(10) NOT NULL,
   ‘attacker’ varchar(100) NOT NULL,
   ‘signaler’ varchar(100) NOT NULL,
   ‘reason’ varchar(100) NOT NULL,
   ‘time’ double NOT NULL )
```

A.4 Struttura e composizione applicazione

Dopo aver mostrato nel precedente paragrafo la struttura della base di dati utilizzata, chiudiamo questo appendice dedicato al programmatore mostrando graficamente il diagramma UML dell'intero progetto che evidenzia la "composizione" dell'applicazione SBX ed offre la possibilità di osservare con maggiore chiarezza la costruzione dei singoli blocchi che la definiscono.

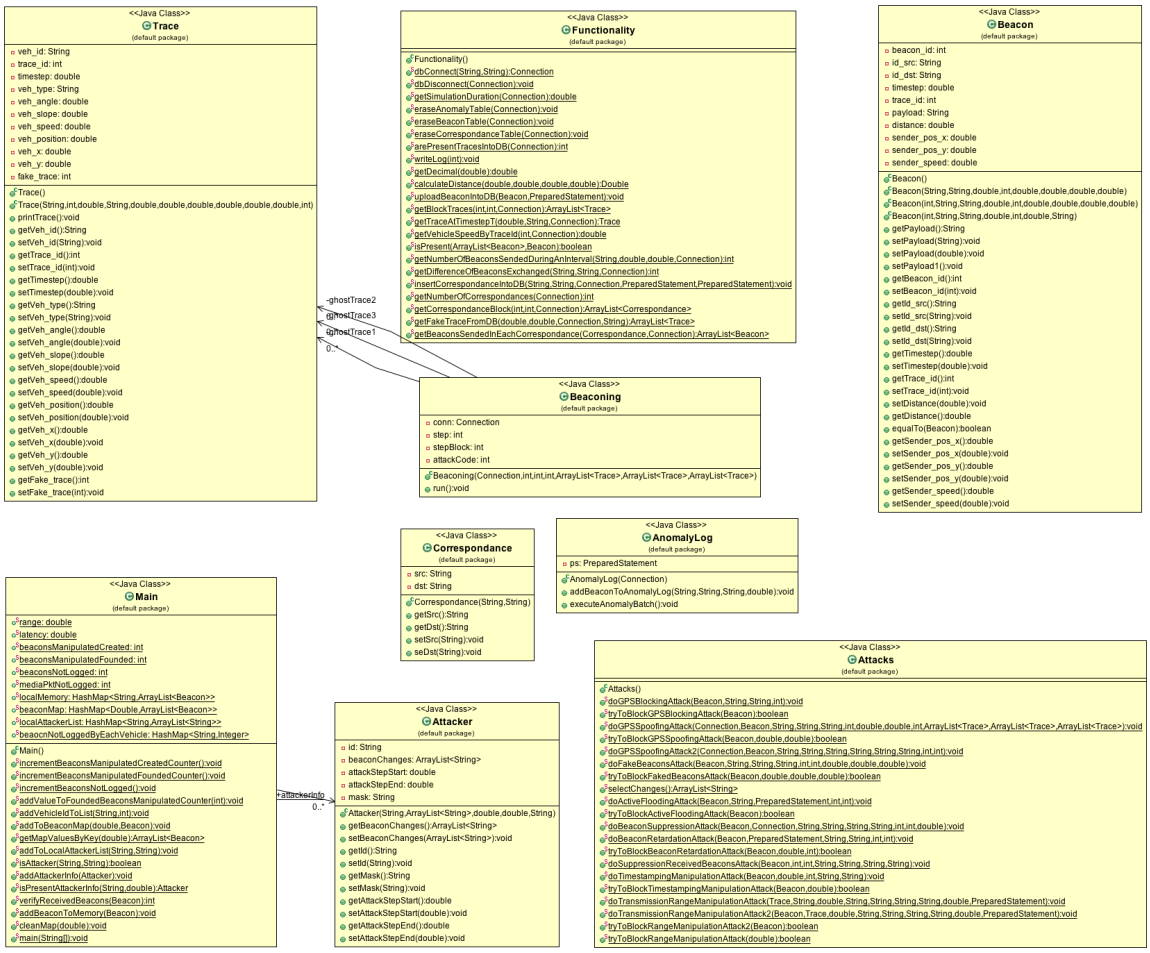


Figura A.1: Class diagram applicazione

A.5 Raccolta delle tabelle

Proprietà oggetto	Descrizione
beacon_id	Rappresenta l'identificativo univoco del beacon
id_src	Contiene l'identificativo di colui che ha generato il beacon
id_dst	Contiene l'dentificativo del destinatario del messaggio
timestep	Individua l'istante in cui il beacon è stato *generato* dal nodo sorgente
trace_id	Contiene l'identità della traccia generata da SUMO a partire dalla quale è stato creato il beacon
payload	Rappresenta il payload del messaggio ovvero il contenuto del messaggio
distance	Indica la distanza che intercorre al momento della generazione del messaggio tra il sorgente e il destinatario
sender_pos_x	Descrive la posizione geografica del veicolo sorgente nell'istante di creazione del beacon (coordinata x)
sender_pos_y	Descrive la posizione geografica del veicolo sorgente nell'istante di creazione del beacon (coordinata y)
sender_speed	Contiene la velocità del veicolo sorgente nel momento in cui è stato creato il beacon

Tabella A.1: Proprietà dell'oggetto Beacon

Proprietà oggetto	**Descrizione**
veh_id	Rappresenta l'identificativo del veicolo a cui fa riferimento la traccia
trace_id	Contiene l'identificativo della traccia generata da SUMO
veh_type	Descrive la tipologia del veicolo, ad esempio nei casi in cui si tratti di un attaccante esso conterrà *attacker*
timestep	Individua l'istante temporale virtuale a cui si riferisce la traccia
vehicle_angle	Descrive la posizione del veicolo
vehicle_slope	Rappresenta l'angolo di inclinazione del veicolo
vehicle_speed	Descrive la velocità del veicolo nell'istante di cattura della traccia
vehicle_position	Contiene la descrizione della posizione nella quale si trova il veicolo (ad esempio l'identificativo della strada o della corsia)
veh_x	Descrive la posizione geografica del veicolo nell'istante di cattura (coordinata x)
veh_y	Descrive la posizione geografica del veicolo nell'istante di cattura (coordinata y)
fake_trace	Identifica la natura della traccia (0 se reale, 1 se fake)

Tabella A.2: Proprietà dell'oggetto Traccia

Proprietà oggetto	Descrizione
id_correspondance	Rappresenta l'identificativo univoco della corrispondenza tra due nodi
id_vehicle_1	Contiene l'identificativo del nodo che ha generato il beacon
id_vehicle_2	Contiene l'identificativo del nodo che ha ricevuto il beacon

Tabella A.3: Proprietà dell'oggetto Correspondance

Proprietà oggetto	Descrizione
anomaly_id	Rappresenta l'identificativo dell'anomalia
attacker	Rappresenta l'identificativo del nodo che è stato etichettato come *attaccante*
signaler	Rappresenta l'identificativo del veicolo *spia* che ha segnalato l'attacco
reason	Comprende la motivazione che ha portato alla segnalazione dell'anomalia
time	Indica l'istante temporale (`timestep`) in cui è stata rilevata l'anomalia

Tabella A.4: Proprietà dell'oggetto Anomaly

Codice	**Attacco**
0	Nessun attacco
1	GPS blocking
2	GPS spoofing (modalità 1)
3	GPS spoofing (modalità 2)
4	Fake beacon generation
5	Active flooding
6	Beacon suppression (modalità 1)
7	Beacon suppression (modalità 2)
8	Beacon suppression (modalità 3)
9	Beacon retardation
10	Beacon received suppression (modalità 1)
11	Beacon received suppression (modalità 2)
12	Timestamping manipulation
13	Transmission range manipulation

Tabella A.5: Codici degli attacchi

Appendice B

Manuale utente

L'esecuzione dell'applicazione sviluppata e descritta finora con tutte le sue proprietà e con lo script presentato nella sezione precedente è stata testata in un ambiente Unix ed in particolare in un sistema MacOS. Per l'esecuzione su altri sistemi si consiglia semplicemente di agire sullo script al fine di utilizzare le chiamate adatte e consentire una corretta esecuzione delle funzioni ivi definite.

B.1 Installazione di SUMO

Uno dei tool fondamentali per la preparazione di un ambiente virtuale sul quale poter eseguire i test per lo sviluppo della varie funzionalità di una rete VANET è senza dubbio rappresentato dal simulatore di traffico e nello specifico caso da SUMO. Nel presente lavoro di tesi è stato utilizzata la versione 0.30.0 del suddetto simulatore che tra le altre cose offre anche un'interfaccia grafica grazie alla quale è possibile visualizzare in tempo reale gli spostamenti dei veicoli sullo scenario simulato datogli in pasto tramite il caricamento del file con estensione *.sumo.cfg*. La versione più recente del *Simulator of Urban MObility*, alias SUMO, può essere scaricata al seguente link mentre per un'installazione completa si consiglia fortemente di seguire questa documentazione che sarà capace di offrire anche all'utente meno esperto un ottimo supporto per una corretta configurazione sul proprio sistema di questo tool fondamentale.

B.2 Scenario della simulazione

Le simulazioni eseguite con SUMO sono state ottenute grazie all'utilizzo di uno scenario cittadino predefinito nato dal progetto di ricerca iTetris che può essere liberamente scaricato al seguente link. Esso riproduce fedelmente una parte della Città di Bologna anche se bisogna dire che il presente lavoro di tesi ha richiesto una modifica opportuna dei vari file che compongono il pacchetto e che descrivono l'ambiente affinché potesse essere definita una duplice configurazione che permettesse di simulare sia gli spostamenti dei veicoli autorizzati che dei veicoli attaccanti

all'interno dello scenario virtuale creato. Poichè questa configurazione è indispensabile per l'esecuzione delle simulazioni qui presentate e implementate all'interno dell'applicazione SBX ho deciso di renderla disponibile attraverso il seguente link al mio cloud personale. Inoltre, se lo si ritiene opportuno, è possibile apportare delle ulteriori modifiche al preset appena indicato andando ad agire sui file con estensione *.sumo.cfg* presenti al suo interno per modificare ad esempio la durata della simulazione mediante il settaggio dei nuovi valori alle voci `begin value` e `end value` presenti all'interno del tag `<time>`.

B.3 Variazione dello scenario cittadino

Come detto più volte lo scenario cittadino di base utilizzato per la simulazione con SUMO rappresenta in scala una parte della città di Bologna. Tuttavia l'utente potrebbe decidere di utilizzare uno scenario alternativo, ad esempio quello di Tapas Cologne che può essere scaricato al presente link o altre configurazioni che è possibile trovare in rete. Di seguito è mostrata passo passo la procedura da seguire per fare questa operazione prendendo come esempio il pacchetto Tapas Cologne.

Una volta scaricato il pacchetto bisognerà estrarlo e modificarlo poiché non bisogna dimenticare che è necessario inserire i veicoli attaccanti per eseguire gli attacchi. Il mio consiglio per svolgere questo lavoro nel modo più semplice ed immediato possibile è quello di creare una prima cartella chiamata "real" all'interno della quale estrarre la configurazione originale e una seconda cartella, chiamata ad esempio "fake", nella quale andare inizialmente ad estrarre la stessa configurazione che dovrà però essere modificata perché sarà quella che ci permetterà di inserire i veicoli attaccanti all'interno dello scenario simulato. Una volta estratto il contenuto dello zip è possibile notare la presenza di una serie di file XML. Procediamo quindi con la modifica della configurazione "fake": accedere alla cartella "fake" e modificare i file con estensione .rou.xml e .sumo.cfg. Onde evitare di doversi creare delle rotte personalizzate per i soli veicoli attaccanti è sufficiente accedere in scrittura al file .rou.xml e modificarlo. Questo file contiene le rotte di tutti i veicoli che prendono parte alla simulazione. L'idea è quella di creare un altro file che chiamiamo fake.rou.xml che conterrà la stessa intestazione dell'originale (basta copiare la parte iniziale del file) ma rotte differenti. I veicoli attaccanti dovranno essere scelti tra quelli presenti all'interno del tag <routes>. E' sufficiente prelevare le rotte dei veicoli che vogliamo far diventare degli attaccanti e copiarli sul file fake.rou.xml facendo attenzione ad eliminarli dal file originale altrimenti durante l'esecuzione della simulazione ci ritroveremo ad aver due veicoli con la stessa identità che compiono lo stesso percorso. Volendo, per completezza, è possibile modificare il "type" dei veicoli inseriti sul file fake.rou.xml ed inserire l'etichetta "attacker". Se si decidesse di apportare questa modifica al tipo di veicolo bisogna modificare anche le voci presenti all'interno del seguente tag (lasciare inalterate quelle scritte in maiuscolo):

```
<vType id=‘‘attacker’’ length=LEN minGap=GAP maxSpeed=SPEED
   vClass=‘‘attacker’’ guiShape=SHAPE color=‘‘red’’/>
```

Talvolta è possibile che le proprietà dei veicoli siano definite all'interno di un file esterno con estensione .add.xml. E' il caso dello scenario che ho utilizzato per

le mie simulazioni. Nel caso in cui si dovesse trovare una situazione del genere basta accedere al file da me editato e valutare le modifiche apportate che saranno sostanzialmente uguali a quelli appena indicati all'interno del tag <vType>. In questo modo è stata definita una nuova classe di utenti attacker che, spostandosi sulla rete, ci consentiranno di simulare gli attacchi nei confronti dei nodi buoni che saranno, come ormai abbiamo capito, rappresentati da tutti i veicoli generati dalla simulazione "real".

Una volta creato il file fake.rou.xml e aggiornato l'originale, bisogna copiare quest'ultimo nella cartella "real" andando quindi a sovrascrivere il vecchio file avente estensione .rou.xml. A questo punto è necessario modificare nella cartella "fake" il file .sumo.cfg. Questo rappresenta il file di configurazione che sarà dato in pasto a SUMO e che ci permetterà di eseguire la simulazione ma prima di far ciò rinominiamo tutti i file all'interno della cartella "fake" con un nome che ci indichi il tipo di configurazione, ad esempio "attack" o "fake". Fatto ciò procediamo con la modifica del file .sumo.cfg. Questo avrà un tag <input> all'interno del quale vengono indicati tutti i file esterni che dovranno essere importati ed utilizzati da SUMO. Fare attenzione ad inserire i nomi in modo corretto altrimenti ci verrà generato un errore in fase di lancio della simulazione. I tag <begin value> e <end value> indicano rispettivamente l'istante di inizio e fine (in secondi) della simulazione e possono essere impostati a piacimento. All'interno del tag <time> bisognerà aggiungere la direttiva `<step-length value=‘‘0.1’’/>` indispensabile per comunicare a SUMO la frequenza di generazione delle tracce e quindi la definizione del cosiddetto timestep che noi vogliamo sia pari a un decimo di secondo.

A questo punto abbiamo ottenuto due diverse configurazioni "real" e "fake" che possono essere eseguite con SUMO. Adesso è sufficiente creare una cartella unica contenente tutti i file dell'una e dell'altra configurazione, facendo attenzione al fatto che non vi siano conflitti. Se si volesse riutilizzare lo script è sufficiente copiarlo all'interno di questa cartella ed aggiornarlo inserendovi i nuovi nomi delle configurazioni appena create che si vogliono eseguire nelle funzioni ivi definite per la creazione delle tracce da salvare poi sulla base di dati che saranno indispensabili all'applicazione SBX per svolgere il proprio compito.

B.4 Installazione di MySQL Community Server

Tra gli strumenti utilizzati si è ritenuta indispensabile una base di dati sulla quale andare a salvare i dati prodotti dalle elaborazioni effettuate nei vari step del processo e dalla quale poter prelevare gli stessi nei momenti in cui si riterranno utili per uno specifico scopo. La base di dati utilizzata in questo caso è una delle più diffuse al mondo ed è rilasciata con licenza open source: MySQL. Questa essendo ampiamente diffusa e supportata può essere facilmente scaricata nella versione più adatta al proprio sistema dalla seguente pagina ufficiale e successivamente installata seguendo se fosse il caso le istruzioni e i consigli per una corretta installazione nella guida presente al seguente link. Una volta conclusa l'installazione si avrà un ambiente quasi pronto e un sistema quasi funzionante al quale però manca ancora il passaggio finale relativo alla configurazione dell'accesso al server. Naturalmente è

compito dell'utilizzatore finale decidere e scegliere le credenziali di accesso personali che gli consentiranno poi di accedere in totale sicurezza alle proprie informazioni. Per qualsiasi dubbio o incertezza è possibile affidarsi alla ottima documentazione presente in questa pagina. Le credenziali sono di assoluta importanza in quanto serviranno anche allo script *RunProcess.sh* e all'applicazione SBX per svolgere il proprio lavoro. Una volta configurato l'accesso al server MySQL è sufficiente seguire i seguenti passaggi per poter portare a termine la configurazione dell'ambiente virtuale al fine di una corretta esecuzione dell'intera applicazione:

1. Aprire il browser web e digitare nella barra di ricerca *http://localhost/phpmyadmin/*
2. Digitare negli appositi form le credenziali personali precedentemente configurate e cliccare su *Esegui*
3. Creare una nuova base di dati
4. Selezionare il tab *Importa* dalla barra in alto
5. Selezionare "*sfoglia...*" e una volta selezionata la cartella scaricata in precedenza contenente tutti i dati dell'applicazione, scegliere il file rinominato "*tesi_db.sql*"

A questo punto se tutto è andato a buon fine si otterrà la configurazione richiesta per una corretta esecuzione del progetto qui sviluppato altrimenti se qualcosa non dovesse funzionare si consiglia di ripetere nuovamente la procedura appena descritta facendo attenzione ad eseguire correttamente tutti i passaggi indicati.

B.5 Esecuzione dell'applicazione

Conclusa la fase di configurazione dell'ambiente è arrivato il momento di focalizzare l'attenzione sull'applicazione vera e propria anche se prima che si possa procedere con l'esecuzione è indispensabile accedere in scrittura allo script *RunProcess.sh* per modificarne alcuni parametri. In particolare bisognerà configurare secondo le proprie esigenze le seguenti variabili:

- `_db` deve contenere il nome che è stato dato alla propria base di dati;
- `_db_user` deve contenere lo username scelto per l'accesso alla base di dati;
- `_db_password` deve contenere la password associata allo username indicato al passaggio precedente;
- `_thread_num` deve specificare il numero di thread che vogliono essere lanciati in parallelo;
- `_range` deve indicare il range di trasmissione che deve essere utilizzato per il processo di Beaconing (valore standard 300 metri);

- `_attack_code` deve specificare il codice dell'attacco che si vuol simulare (per i codici degli attachi fare riferimento alla Tabella A.5 presente nell'Appendice A);
- `_simulation_duration` deve specificare la durata della simulazione che si vuol lanciare con l'applicazione SBX (in secondi).

A questo punto è sufficiente salvare lo script appena modificato sovrascrivendo la vecchia configurazione e cliccarci su per avviarne l'esecuzione. Fatto ciò l'applicazione inizierà il suo lavoro e una volta terminata verrà generato un file di log all'interno della directory nella quale risiede lo script prima citato dal nome *LogResult* che conterrà un resoconto dell'operazione appena eseguita ed in particolare segnalerà, nel caso in cui si fosse deciso di simulare un attacco, tutte le informazioni relative al numero di beacon generati dagli attaccanti e immessi sulla rete e al numero di beacon catturati dal relativo sistema di rilevazione appositamente sviluppato per la protezione della rete da quel particolare attacco. Ad ogni esecuzione l'applicazione si preoccuperà di ripulire totalmente le tabelle della base di dati che dovranno essere popolate di volta in volta seguendo una strategia sempre diversa che varii al variare della tipologia di attacco che si è scelto di simulare. Inoltre se si vuole evitare di eseguire una nuova simulazione con SUMO ogni volta che si decide di lanciare/simulare un attacco, è sufficiente accedere in scrittura allo script e commentarne con un # le righe nella parte finale che eseguono le chiamate alle funzioni *generateRealTrace()* e *generateFakeTrace()*. In questo modo sarà possibile generare le tracce con SUMO una sola volta ed evitare di ricrearle ogni volta che si vuol avviare una nuova simulazione ottenendo così un risparmio in termini di tempo di esecuzione dell'intero processo. Naturalmente se si dovesse scegliere questa "scorciatoia" bisognerà lasciare scommentata sullo script la riga che si preoccupa di lanciare l'applicazione SBX ma soprattutto sarà necessario che il database sia già stato popolato con le tracce altrimenti l'applicazione SBX, che al lancio effetua un check dello stato della base di dati, ritornerà un errore che segnalerà appunto l'impossibilità di procedere con la simulazione dell'attacco scelto e individuato tramite il codice datole in ingresso. Se tutti i controlli sono andati a buon fine allora l'applicazione verrà eseguita normalmente fornendo alla fine il resoconto dell'operazione svolta.

Bibliografia

[1] J. Kao, Research University Group Interests, http://www.cs.nthu.edu.tw/~jungchuk/research.html

[2] S. Al-Sultan, M. M. Al-Doori, A. H. Al-Bayatti, H. Zedan, "A Comprehensive Survey on Vehicular Ad Hoc Network", Journal of Network and Computer Applications, Vol. 37, January 2014, pp. 380-392, DOI 10.1016/j.jnca.2013.02.036

[3] R. Fracchia, M. Meo, D. Rossi, "VANETs: To Beacon or not to Beacon?", Proceedings of the Autonet at IEEE Globecom, San Francisco (USA), November-December 2006

[4] H. T. Friis, "A Note on a Simple Transmission Formula", Proceedings of the Institute of Radio Engineers, Vol. 34, May 1946, pp. 254-256, DOI 10.1109/JRPROC.1946.234568

[5] A. M. S. Abdelgader, W. Lenan, "The Physical Layer of the IEEE 802.11p WAVE Communication Standard: The Specifications and Challenges", Proceedings of the World Congress on Engineering and Computer Science, San Francisco (USA), Vol. 2, October 2014, pp. 22-24

[6] SAE International, "SAE J2735 Dedicated Short Range Communications (DSRC) Message Set Dictionary", RFC-J2735_200911, http://standards.sae.org/, November 2009

[7] H. El Ajaltouni, R. W. Pazzi, A. Boukerche, "An Efficient QoS MAC for IEEE 802.11p Over Cognitive Multichannel Vehicular Networks", IEEE International Conference on Communications (ICC), Ottawa (Canada), June 10-15, 2012, pp. 413-417, DOI 10.1109/ICC.2012.6364353

[8] W. Peng, X. Lu, "On the Reduction of Broadcast Redundancy in Mobile Ad Hoc Networks", Proceedings of the 1st ACM International Symposium on Mobile Ad Hoc Networking & Computing, Boston (USA), August 11, 2000, pp. 129-130, DOI 10.1109/MOBHOC.2000.869221

[9] Y.C. Tseng, S.Y. Ni, "The Broadcast Storm Problem in a Mobile Ad Hoc Network", Wireless Networks, Vol. 8, No. 2, March 2002, pp. 153-167, DOI 10.1023/A:1013763825347

[10] F. Li, Y. Wang, "Routing in Vehicular Ad hoc Networks: A survey", IEEE Vehicular Technology Magazine, Vol. 2, No. 2, June 2007, pp. 12-22, DOI 10.1109/MVT.2007.912927

[11] B. Paul, Md. Ibrahim, Md. A. N. Bikas, "VANET Routing Protocols: Pros and Cons", International Journal of Computer Applications, Vol. 20, No. 3, April 2011, pp. 28-34, DOI 10.5120/2413-3224

[12] K. C. Lee, U. Lee, M. Gerla, "Survey of Routing Protocols in Vehicular Ad Hoc Networks" nel libro "Advances in Vehicular Ad-Hoc Networks: Developments and Challenges", M. Watfa, 2010, pp. 149-170, DOI 10.4018/978-1-61520-913-2.ch008

[13] H. B. Safdar, K. Gwanghyeon, H. A. Syed, K. Dongkyun, "Hybrid Adaptive Beaconing in Vehicular Ad Hoc Networks: A Survey", International Journal of Distributed Sensor Networks, Vol. 11, No. 5, January 2015, pp. 1-16, DOI 10.1155/2015/390360

[14] V. Nundloll, G. Blair, P. Grace, "A Component-Based Approach for (Re)-Configurable Routing in VANETs", Proceedings of the 8th International Workshop on Adaptive and Reflective Middleware, Urbana Champaign (USA), December 01, 2009, pp. 1-6, DOI 10.1145/1658185.1658187

[15] G. Fenu, M. Nitti, "Strategies to Carry and Forward Packets in VANET" nel libro "Digital Information and Communication Technology and Its Applications. DICTAP 2011. Communications in Computer and Information Science" a cura di H. Cherifi, J. S. Zain, E. El-Qawasmeh, Springer, 2011, pp. 662-674, DOI 10.1007/978-3-642-21984-9_54

[16] K. Katsaros, M. Dianati, "Effective Implementation of Location Services for VANETs in Hybrid Network Infrastructures", IEEE International Conference on Communications Workshops (ICC), Budapest (Hungary), June 9-13, 2013, pp. 521-525, DOI 10.1109/ICCW.2013.6649289

[17] H. Saleet, O. Basir, R. Langar, R. Boutaba, "Region-Based Location-Service-Management Protocol for VANETs", IEEE Transactions on Vehicular Technology, Vol. 59, No. 2, February 2010, pp. 917-931, DOI 10.1109/TVT.2009.2033079

[18] Y. J. Chang, T. L. Shih, "Intersection Location Service and Performance Comparison of Three Location Service Algorithms for Vehicular Ad Hoc Networks in City Environments", Wireless Pervasive Computing, ISWPC, Santorini (Greece), May 7-9, 2008, pp. 562-565, DOI 10.1109/ISWPC.2008.4556271

[19] X.-Y. Bai, X.-M. Ye, J. Li, H. Jiang, "VLS: A Map-Based Vehicle Location Service for City Environment", IEEE International Conference on Communications Workshops (ICC), Dresden (Germany), June 14-18, 2009, pp. 1-5, DOI 10.1109/ICC.2009.5199568

[20] M. Boussedjra, J. Mouzna, P. Bangera, M. M. M. Pai, "Map-Based Location Service for VANET", International Conference on Ultra Modern Telecommunications & Workshops, St. Petersburg (Russia), October 12-14, 2009, pp. 1-6, DOI 10.1109/ICUMT.2009.5345470

[21] LinkBird-MX Product Manual, http://www.nec.co.jp/press/en/0811/images/1301-01.pdf

[22] A. Festag, R. Baldessari, W. Zhang, L. Le, "CAR-2-X Communication SDK - A Software Toolkit for Rapid Application Development and Experimentations", IEEE Vehicular Networking and Applications Workshop, Dresden (Germany), June 14-18, 2009, pp. 1-5, DOI 10.1109/ICCW.2009.5208070

[23] OMNeT++ Home Page, http://www.omnetpp.org/

[24] S. Mccanne, S. Floyd, K. Fall, ns2 (Network Simulator 2), https://www.isi.edu/nsnam/ns/

[25] D. Krajzewicz, J. Erdmann, M. Behrisch, L. Bieker, "Recent Development and Applications of SUMO - Simulation of Urban MObility", International Journal On Advances in Systems and Measurements, Vol. 5, December 2012, pp. 128-138

[26] Main page VABENE project, http://verkehrsforschung.dlr.de/de/projekte/vabene

[27] Main page VEU project, http://verkehrsforschung.dlr.de/projekte/veu

[28] Installing SUMO, http://sumo-sim.org/userdoc/Installing/Binary.html

[29] Open Street Map Official Site, http://www.openstreetmap.org/

[30] M. van Eenennaam, W. K. Wolterink, G. Karagiannis, G. Heijenk, "Exploring the Solution Space of Beaconing in VANETs", Proceedings of IEEE Vehicular Networking Conference (VNC), Tokyo (Japan), October 28-30, 2010, pp. 1-8, DOI 10.1109/VNC.2009.5416370

[31] S. Djahel, Y. Ghamri-Doudane, "A Robust Congestion Control Scheme for Fast and Reliable Dissemination of Safety Messages in VANETs", Proceeding of the 2012 IEEE Conference Wireless Communications and Networking, Paris (France), April 1-4, 2012, pp. 2264-2269, DOI 10.1109/WCNC.2012.6214170

[32] P. Papadimitratos, A. Jovanovic, "Protection and Fundamental Vulnerability of GNSS", International Workshop on Satellite and Space Communications (IWSSC), Touluse (France), October 1-3, 2008, pp. 167-171, DOI 10.1109/IWSSC.2008.4656777

[33] W. Zhang, A. Festag, R. Baldessari, L. Le, "Congestion Control for Safety Messages in VANETs: Concepts and Framework", 8th International Conference on ITS Telecommunications, Phuket (Thailand), October 24, 2008, pp. 199-203, DOI 10.1109/ITST.2008.4740256

[34] G. Caizzone, P. Giacomazzi, L. Musumeci, G. Verticale, "A Power Control Algorithm with High Channel Availability for Vehicular Ad Hoc Networks", IEEE International Conference on Communications (ICC), Seoul (Korea), May 16-20, 2005, pp. 3171-3176, DOI 10.1109/ICC.2005.1494999

[35] C. Thaina, K. Na Nakorn, K. Roviboonchai, "A Study of Adaptive Beacon Transmission on Vehicular Ad-Hoc Networks", IEEE 13th International Conference on Communication Technology, Jinan (China), September 25-28, 2011, pp. 597-602, DOI 10.1109/ICCT.2011.6157946

[36] T. Settawatcharawanit, S. Choochaisri, C. Intanagonwiwat, K. Rojviboochai, "V-DESYNC: Desynchronization for Beacon Broadcasting on Vehicular Networks", IEEE 75th Vehicular Technology Conference (VTC Spring), Yokohama (Japan), May 6-9, 2012, pp. 1-5, DOI 10.1109/VETECS.2012.6239966

[37] M. Slavik, I. Mahgoub, M. M. Alwakeel, "Analysis of Beaconing Message Rate in VANET Multi-Hop Broadcast Protocols", 9th International Conference on High Capacity Optical Networks and Enabling Technologies (HONET), Istanbul (Turkey), December 12-14, 2012, pp. 37-41, DOI 10.1109/HONET.2012.6421431

[38] K. Na Nakorn, Y. Ji, K. Rojviboonchai, "Bloom Filter for Fixed-Size Beacon in VANET", Proceedings of the IEEE 79th Vehicular Technology Conference, Seoul (Korea), May 18-21, 2014, pp. 1-5, DOI 10.1109/VTCSpring.2014.7022849

[39] H. Noori, B. B. Olyaei, "A Novel Study on Beaconing for VANET-Based Vehicle to Vehicle Communication: Probability of Beacon Delivery in Realistic Large-Scale Urban Area using 802.11p", International Conference on Smart Communications in Network Technologies (SaCoNeT), Paris (France), June 17-19, 2013, pp. 1-6, DOI 10.1109/SaCoNeT.2013.6654589

[40] J.-P. Hubaux, S. Capkun, J. Luo, "The Security and Privacy of Smart Vehicles", IEEE Security and Privacy, Vol. 2, No. 3, June 2004, pp. 49-55, DOI 10.1109/MSP.2004.26

[41] K. Sampigethaya, L. Huang, M. Li, R. Poovendran, K. Matsuura, K. Sezaki, "CARAVAN: Providing Location Privacy for VANET", Proceedings of the Workshop on Embedded Security in Cars (ESCAR), Cologne (Germany), November 29-30, 2005

[42] P. Golle, D. Greene, J. Staddon, "Detecting and Correcting Malicious Data in VANETs", Proceedings of the 1st ACM International Workshop on Vehicular Ad Hoc Networks, Philadelphia (USA), October 01, 2004, pp. 29-37, DOI 10.1145/1023875.1023881

[43] IEEE Std P1609.2 Version 1, "Standard for Wireless Access in Vehicular Environments - Security Services for Applications and Management Messages", "Last Revision March 2016", DOI 10.1109/IEEESTD.2016.7426684

[44] M. Raya, D. Jungels, P. Papadimitratos, I. Aad, J.P. Hubaux, "Certificate Revocation in Vehicular Networks", Laboratory for computer Communications and Applications (LCA) School of Computer and Communication Sciences, EPFL, Switzerland, January 2006

[45] S. Farrell, R. Housley, S. Turner, "An Internet Attribute Certificate Profile for Authorization", RFC-5755, January 2010

[46] S. Santesson, M. Nystrom, T. Polk, "Internet X.509 Public Key Infrastructure: Qualified Certificates Profile", RFC-3739, March 2004

[47] T. Moore, M.Raya, J. Clulow, P. Papadimitratos, R. Anderson, J-P. Hubaux, "Fast Exclusion of Errant Devices from Vehicular Networks", 5th Annual IEEE Communications Society Conference on Sensor, Mesh and Ad Hoc Communications and Networks, San Francisco (USA), June 16-20, 2008, pp. 135-143, DOI 10.1109/SAHCN.2008.26

[48] I. Enrici, M. Ancilli, A. Lioy, "A psychological approach to information technology security", HSI-2010: 3rd Int. Conf. on Human System Interactions, Rzeszòw (Poland), May 13-15, 2010, pp. 459-466, DOI 10.1109/HSI.2010.5514528

[49] I. Ahmed, I. Ahmad, H. Hasbullah, Jamalul-lail bin Ab Manan, "Classes of Attacks in VANET", 2011 Saudi International Electronics, Communications and Photonics Conference (SIECPC), Riyadh (Saudi Arabia), April 24-26, 2011, pp. 1-5, DOI 10.1109/SIECPC.2011.5876939

[50] I. A. Sumra, H. B. Hasbullah, J. B. AbManan, "Attacks on Security Goals (Confidentiality, Integrity, Availability) in VANET: A Survey" nel libro "Vehicular Ad Hoc Networks for Smart Cities: First International Workshop 2014" a cura di A. Laouiti, A. Qayyum, M. N. Mohamad Saad, Springer, 2015, pp. 51-61, DOI 10.1007/978-981-287-158-9_5

[51] K. Verma, H. Hasbullah, A. Kumar, "Prevention of DoS Attacks in VANET", Wireless Personal Communications, Vol. 73, No. 1, November 2013, pp. 95-126,

DOI 10.1007/s11277-013-1161-5

[52] Y. C. Hu, A. Perrig, D. B. Johnson, "Ariadne: A Secure On-Demand Routing Protocol for Ad Hoc Networks", Wireless Networks, Vol. 11, No. 1, January 2005, pp. 21-38, DOI 10.1007/s11276-004-4744-y

[53] S. Bittl, A. A. Gonzalez, M. Myrtus, H. Beckmann, S. Sailer, B. Eissfeller, "Emerging Attacks on VANET Security Based on GPS Time Spoofing", IEEE Cyber and Network Security Conference (CNS), Florence (Italy), September 28-30, 2015, pp. 344-352, DOI 10.1109/CNS.2015.7346845

[54] K. Sanzgiri, B. Dahill, B. N. Levine, C. Shields, E. M. Belding-Royer, "A Secure Routing Protocol for Ad Hoc Networks", Proceedings of the 10th IEEE International Conference on Network Protocols (ICNP), Washington (USA), November 12-15, 2002, pp. 78-89, DOI 10.1109/HSI.2010.5514528

[55] I. Dacosta, S. Chakradeo, M. Ahamad, P. Traynor, "One-Time Cookies: Preventing Session Hijacking Attacks with Stateless Authentication Tokens", ACM Transactions on Internet Technology (TOIT), Vol. 12, No. 1, June 2012, pp. 1-24, DOI 10.1145/2220352.2220353

[56] Y. Zhang, W. Lee, "Intrusion Detection in Wireless Ad-Hoc Networks", Proceedings of the 6th Annual International Conference on Mobile Computing and Networking (MobiCom), Boston (USA), August 06-11, 2000, pp. 275-283, DOI 10.1145/345910.345958

[57] S. Gwalani, K. Srinivasan, G. Vigna, E. Belding-Royer, R. Kemmerer, "An Intrusion Detection Tool for AODV-based Ad Hoc Wireless Networks", 20th Annual Computer Security Applications Conference, Tucson (USA), December 06-10, 2004, pp. 16-27, DOI 10.1109/CSAC.2004.6

[58] SEVECOM - Secure Vehicle Communications Project, http://www.sevecom.org/

[59] E. Schoch, F. Kargl, T. Leinmuller, S. Schlott, P. Papadimitratos, "Impact of Pseudonym Changes on Geographic Routing in VANETs" nel libro "Security and Privacy in Ad-Hoc and Sensor Networks: Third European Workshop, ESAS 2006, Hamburg, Germany, September 20-21, 2006, Revised Selected Papers" a cura di L. Buttyán, V. D. Gligor, D. Wesyhoff, Springer Berlin Heidelberg, 2006, pp. 43-57, DOI 10.1007/11964254_6

[60] G. Calandriello, P. Papadimitratos, J-P. Hubaux, A. Lioy, "Efficient and Robust Pseudonymous Authentication in VANET", Proceedings of the 4th ACM International Workshop on Vehicular Ad hoc Networks, Montreal (Canada), September 10, 2007, pp. 19-28, DOI 10.1145/1287748.1287752

[61] POLYCONVERT Manual, http://sumo-sim.org/userdoc/POLYCONVERT.html

[62] DFROUTER Manual, http://sumo-sim.org/userdoc/DFROUTER.html

[63] iTETRIS Research Project, Official Site, http://www.ict-itetris.eu/

[64] TAPAS Cologne Scenario, http://sumo-sim.org/userdoc/Data/Scenarios/TAPASCologne.html

[65] SUMO Output, http://sumo-sim.org/userdoc/Simulation/Output.html

[66] TraceExporter, Manual, http://www.sumo.dlr.de/userdoc/Tools/TraceExporter.html

[67] xml2csv, Manual, http://sumo.dlr.de/wiki/Tools/Xml#xml2csv.py

[68] S. Bittl, A. A. Gonzalez, M. Myrtus, H. Beckmann, S. Sailer, B. Eissfeller, "Emerging Attacks on VANET Security Based on GPS Time Spoofing", IEEE Conference on Communications and Network Security (CNS), Florence (Italy), September 28-30, 2015, pp. 344-352, DOI 10.1109/CNS.2015.7346845

[69] R. Panayappan, J. Trivedi, A. Studer, A. Perrig, "VANET-based Approach for Parking Space Availability", Proceedings of the 4th ACM International Workshop on Vehicular Ad Hoc Networks (VANET 2007), Montreal (Canada), September 10, 2007, pp. 75-76, DOI 10.1145/1287748.1287763

[70] N. Tippenhauer, C. Pöpper, K.Rasmussen, S. Capkun, "On the Requirements for Successful GPS Spoofing Attacks", Proceedings of the 18th ACM Conference on Computer and Communication Security, Chicago (USA), October 17-21, 2011, pp. 75-86, DOI 10.1145/2046707.2046719

[71] E. Schoch, F. Kargl, "On the Efficiency of Secure Beaconing in VANETs", Proceedings of the 3rd ACM Conference on Wireless Network Security (WiSec 2010), New York (USA), March 22-24, 2010, pp. 111-116, DOI 10.1145/1741866.1741885

[72] P. Papadimitratos, G. Calandriello, A. Lioy, J-P. Hubaux, "Impact of Vehicular Communication Security on Transportation Safety", Proceedings of the 28th IEEE Conference on Computer Communications (INFOCOM) Workshop on Mobile Networking for Vehicular Environments (MOVE), Phoenix (USA), April 13-18, 2008, pp. 1-6, DOI 10.1109/INFOCOM.2008.4544663

[73] U.Rajput, F. Abbas, H. Eun, R. Hussain, H.Oh, "Two Level Privacy Preserving Pseudonymous Authentication Protocol for VANET", IEEE 11th International Conference on Wireless and Mobile Computing, Networking and Communications (WiMob), Abu Dhabi (United Arab Emirates), October 19-21, 2015, pp. 643-650, DOI 10.1109/WiMOB.2015.7348023

[74] G. Karagiannis, R. Wakikawa, J. Kenny, C. J. Bernardos, F. Kargl, "Traffic safety applications requirements", IETF Draft, 2010, `http://tools.ietf.org/html/draft-karagiannis-traffic-sfety-requirements-02`