UNIMORE
UNIVERSITÀ DEGLI STUDI DI
MODENA E REGGIO EMILIA

Dipartimento di Scienze Fisiche,Informatiche e Matematiche

Corso di Laurea Magistrale in Informatica

Framework per la valutazione degli impatti di attacchi V2V in reti veicolari

Relatore:
Prof. Mirco Marchetti

Candidato:
Salvatore Bianco

Correlatore:
Ing. Giovanni Gambigliani Zoccoli

Anno Accademico 2024/2025

Indice

Capitolo 1

Introduzione

In questo capitolo si propongono degli esempi per gli oggetti utilizzati più di frequente in latex: la Sezione 1.1 descrive come scrivere citazioni, la Sezione 1.2 propone degli esempi di oggetti float, la Sezione 1.3 descrive come compilare questo documento.

1.1 Citazioni

Inserisco qualche citazione per mostrare la bibliografia. Per gli articoli accademici è quasi sempre possibile reperire i blocchi da inserire nel file bib da scholar [2], come ad esempio [1]. Scholar in questo caso è una risorsa/sito online e per questo. Precediamo le citazione da uno spazio indivisibile tramite il carattere ~.

1.2 Oggetti float

Nella Sezione 1.2.1 si propone un esempio di figura float, mentre nella Sezione 1.2.2 si propone un esempio di tabella float.

1.2.1 Figure

La Figura 1.1 è un esempio di figura float.

Figura 1.1: Esempio di figura float in latex.

1.2.2 Tabelle

La Tabella 1.1 è un esempio di tabella.

allineamento centrale	allineamento a sinistra	allineamento a destra
centrale	sinistra	destra

Tabella 1.1: Esempio di tabella float in latex.

1.3 Compilazione

Di seguito il codice da utilizzare per generare il pdf:

```
$ pdflatex main.tex
$ bibtex main.aux
$ pdflatex main.tex
$ pdflatex main.tex
```

Capitolo 2

Background

In questo capitolo si descriveranno i framework utilizzati, si tratta dei software più utilizzati per le simulazioni veicolari. Questi programmi svolgono funzioni differenti,ma insieme permettono di analizzare come i veicoli interagiscono tra loro, con le reti e con il mondo, pemettendo quindi la simulazione di scenari di guida complessi.

2.1 SuMo

SuMo (**Simulator of Urban Mobility**) è un framework di simulazione del traffico avanzato, multimodale e interamente open-source. La sua caratteristica distintiva risiede nell'approccio **microscopico**: il software non si limita a modellare i flussi di traffico aggregati, ma simula il comportamento e il movimento di ogni singolo agente (come veicoli o pedoni) all'interno della rete stradale.

Fin dal suo concepimento da parte del DLR (Centro Aerospaziale Tedesco), il progetto è stato guidato da due obiettivi tecnici fondamentali: la **velocità** di esecuzione e la **portabilità**. Per raggiungere queste finalità, SUMO è stato progettato con un'architettura non monolitica, ma **modulare**. L'intero pacchetto software è infatti una "suite" di strumenti e applicazioni separate, ciascuna ottimizzata per un compito specifico.

Un chiaro esempio di questa filosofia è la gestione della domanda di traffico. La generazione dei viaggi che i veicoli devono compiere non è integrata nel motore di simulazione principale, ma viene gestita da strumenti accessori, come lo script **'randomTrips.py'**.

Questo strumento, ad esempio, viene utilizzato in fase di pre-processamento per generare un elenco di viaggi casuali che serviranno poi da input per la simulazione vera e propria.

Questa separazione dei compiti, sebbene possa rendere la configurazione iniziale più articolata rispetto a soluzioni "tutto-in-uno", conferisce al software una grande flessibilità ed efficienza.

Dal punto di vista operativo, le simulazioni di SUMO sono **continue nello spazio** (permettendo movimenti fluidi) e **discrete nel tempo** (avanzando per piccoli intervalli temporali). Per garantire la piena riproducibilità degli esperimenti, il comportamento del simulatore è **deterministico** per impostazione predefinita, anche se l'utente ha la possibilità di introdurre elementi stocastici (casuali) per modellare scenari più realistici.

Oltre alla simulazione pura, una delle funzionalità più potenti di SUMO è la sua capacità di interazione e controllo in **run-time**. Questa è resa possibile da **TraCI** (Traffic Control Interface), un'API che permette a script esterni (ad esempio, scritti in Python) di connettersi alla simulazione mentre è in esecuzione. Tramite TraCI è possibile interrogare in tempo reale lo stato di qualsiasi oggetto della simulazione (come la velocità di un veicolo o lo stato di un semaforo) e inviare comandi per modificarne dinamicamente il comportamento.

In termini di prestazioni, SUMO è ottimizzato per gestire reti stradali di vasta scala, capaci di includere centinaia di migliaia di archi, mantenendo un'elevata velocità di calcolo. I dati di output possono essere raccolti a diversi livelli di dettaglio: a livello di intera rete, di singolo arco stradale, di singolo veicolo o tramite l'impiego di rilevatori virtuali posizionati sulla mappa.

2.1.1 Traci

TraCI, acronimo di Traffic Command Interface, è una delle funzionalità più potenti di SuMo. Si tratta di un'interfaccia client-server basata su protocollo TCP, creata per cosnsentire l'interazione a run-time con la simulazione.

La simulazione avviata con l'opzione di TraCI abilitata, indica a SuMo di agire come server,pertanto esso apre una specifica socket mettendosi in ascolto. Dall'esterno in-

vece, un programma agisce come client connettendosi al socket TCP per stabilire la comunicazione. Non ci sono limiti specifici sugli script e sui linguaggi (python,C++ e JAVA sono quelli più utilizzati), sebbene la libreria client più documentata sia TraciPy, è disponibile anche una in C++. L'interfaccia TraCI permette un controllo granulare su quasi ogni aspetto della simulazione.
Le sue capacità possono essere suddivise in due categorie principali:

- Recupero dei dati: il client può leggere in tempo rale lo stati di qualsiasi oggetto, ad esempio ottenendo la velocità, la posizione, l'accelerazione etc;
- Invio di comandni : il client può scrivere modifche attive nella simulazione. Ad esempio cambiare dinamicamente la rott adi un veicolo, modificare la durata o la sequenza delle fasi di un impianto semforico o generare nuovi veicoli.

2.2 Omnet++

Omnet++ è un framework di simulazione e una libreria software estensibile, modulare e basata su componenti, implementato in C++. Viene utilizzato in molteplici domini , il suo principale impiego è la costruzione di simulatori di rete.

Vien da se specificare che **Omnet++** non è un simulatore di rete pronto all'uso, ma una piattaforma per crearne di nuovi . LA sua polarità in ambito accademico e industriale derica dalla sua archeittura flessibile e dal robusto supporto di strumenti.

L'architettura di OMNeT++ si fonda su una netta separazione tra la **struttura** del modello e la sua **logica** comportamentale, seguendo un approccio gerarchico.

1. **Definizione dei moduli**: I componenti attivi della simulazione sono detti moduli semplici e la logic adi questi moduli è programmata in C++.

2. **Definizione della tipologia**: La struttura del modello , cioè come i moduli osno connessi tra loro , è definita in un linguaggio dichiarativo detto **NED (network description**. Tramite NED i moduli semplci , definiti precendentemente, possono essere assemblati per crare **moduli composti**, che a loro volta posson oessere raggruppati in gerarchie più complesse.

3. **Configurazione (file .ini)** : L'esecuzione della simulazione veine controlalto tramite **omnetpp.ini**. Questo file permette di definire i parametri del modello , come ad esempio il numero di nodi o la velocità di trasmissione.

L'esecuzione di una simulazione può avvenire tramite due interfacce principali:

- **Cmdenv**: un 'interfaccia a riga di comando leggera ed efficiente;
- **Qtenv**: un'interfaccia grafica interattiva che permette di visualizzare la topologia animare lo scambio di messaggi e ispezionare lo stato dei moduli durante l'esecuzione.

Le funzionalità specifiche per un determinato dominio non sono inteegrate nel kernel di **Omnet**++ ma sono fornite da framework di modelli indipendenti.

Tra questi rientra **INET**, un progetto mantenuto dal tema di Omnet++ che fornisce un'implementazione dettagliata e accurata dell'intero stack protocollare internet e altri numerosi protocolli e componenti di rete.

2.3 Veins

Il framework **Veins** è un ambiente open-source pensato per le simulazioni di reti veicolari , in particolare per i casi d'uso legati alla comunicazione inter-veicolare (V2V) e tra veicoli e infrastruttura (V2I), quindi per le comunicazioni (V2X). L'esigenza di studiare reti veicolari (**VANET**) e sistemi di trasporto intelligneti (**ITS**) e la loro interazione ha portato allo sviluppo del framework **Veins** (Vehicles in Network Simulation), che non è quindi un simulatore a sè stante ma un vero framework di co-simulaizone pensato per orchestrare e collegare SuMo e Omnet++ La sua architettura combina i due framework precedentemente definiti in questo modo:

- **SuMo**: per la simulazione del traffico stradale,
- **Omnet**++: per la simulazione della rete ,gestendo l'invio e la ricezione di messaggi, modellando la propagazione utilizzando uno stack di comunicazione veicolare basato su **IEEE 802.11p**.

Veins,infatti, associa ad ogni veicolo di SuMo un nodo di rete in Omnet++. Ad ogni nodo è associato uno stack che include un interfaccia di rete IEEE 802.11p, un protocollo di beaconing e altre applicaizoni. Il meccanismo di co-simulazione è possibile tramite TraCI, introdotta nel capitolo su SuMo. All'interno dell'ambiente Omnet++, Veins esegue un modulo che basato su TCP, che agisce come client TraCI. Il client stabilisce la connessione con SuMo e gestisce il ciclo di sincronizzazione, recuperando i dati aggiornati di ogni veicolo, che vengono utilizzti per aggiornare i corrispondenti nodi mobili corrispondenti all'interno della simulazione di Omnet++. Il canale di comunicazione è bidirezionale, pertanto un evento in Omnet++ (come può essere una collisione V2V o un'istruzione da un'infrastruttura) necessita di influenzare la dinamica del traffico, e il modulo (**TraCIScenarioManager**) invia comandi TraCI a SuMo, comandi che influenza il comportmamento dei veicoli nella simulazione, forzandoli a compiere delle azioni (ad esempio un cambio di rotta o un'accelerazione).

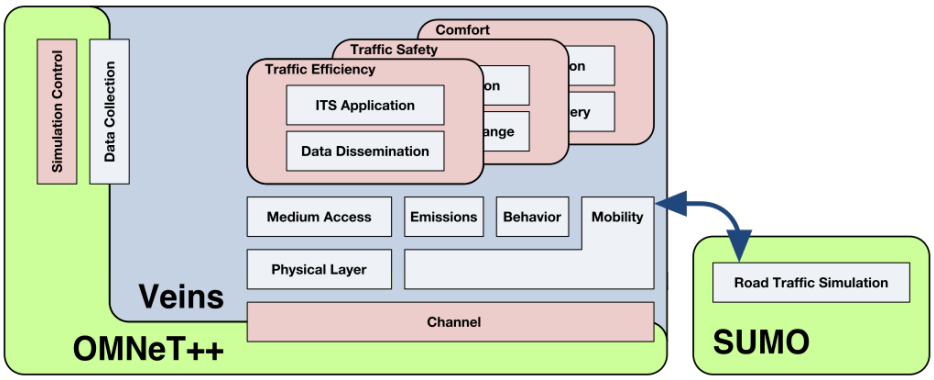

Figura 2.1: Architettura di Co-simulazione di Veins tra OMNeT++ e SUMO. ~[3]

2.4 F^{2}MD

Per analizzare e validare in modo sistematico gli algoritmi di sicurezza nelle reti veicolari, non è sufficiente disporre di un simulatore di rete (**Omnet++**) e di mobilità (**SuMo**). E' necessario un framework aggiuntivo che orchestri gli esperimenti, introduca comportamenti anomali e valuti i risultati. In questo contesto si inserisce **F^{2}MD** (**Framework for Misbehavior Detection**),una piattaforma di simulazione open-source progettata specificamente per lo sviluppo, il test e la comparazione di algoritmi

di rilevamento delle anomalie nei **C-ITS**. Di fonamentale importanza è notare che **F^{2}MD** non è un simulatore a sè stante, ma un'estensione modulare di **Veins**. Esso eredita tutta l'architettura di co-simulazione che permette di far cooperare Omnet++ e SuMo, arrichendola con una suite completa di strumenti specifici per la sicurezza. L'architettura di **F^{2}MD** è organizzata in livelli funzionali, che includono la gestione dei dati di input, il rilevamento delle anomalie (locale e globale) e, di fondamentale importanza, i moduli per l'induzione di comportamenti anomali.

Capitolo 3

Tecnologie

3.1 Vanet

3.2 C-ITS

3.3 V2X

3.4 BSM

Capitolo 4

Implementazione

Capitolo 5

Test

Bibliografia

[1] Uriel Feige, Amos Fiat, and Adi Shamir. Zero-knowledge proofs of identity. *Journal of cryptology*, 1(2):77–94, 1988.

[2] Google. Google scholar. `https://scholar.google.it/`, visited in Sep. 2016.

[3] Veins Team. Veins - the open source vehicular network simulation framework. https://veins.car2x.org/documentation/, 2025.