Slovenská technická univerzita v Bratislave

Fakulta informatiky a informačných technológií

FIIT-5212-102916

# Jakub Ganádik

# Analýza možností výmeny informácií v prostredí navzájom prepojených vozidiel

Bakalárska práca

|  |  |
| --- | --- |
|  |  |

Študijný program: Informatika

Študijný odbor: Informatika

Miesto vypracovania:   
Ústav počítačového inžinierstva a aplikovanej informatiky, FIIT STU, Bratislava

Vedúci práce: Ing. Marek Galinski, PhD.

Máj 2022

*zadná strana titulného listu (prázdna strana)*

# Zadanie práce

Jedným zo základných pilierov inteligentnej dopravy sú navzájom prepojené dopravné prostriedky, či už vozidlá medzi sebou, alebo vozidlá s infraštruktúrou alebo cloudovými službami. Otvoreným a stále aktuálnym problémom zostáva, ako vybrať v konkrétnej situácií vhodnú prístupovú vrstvu a aké informácie medzi sebou vymieňať tak, aby to bolo v prospech celkovej dopravnej situácie. Analyzujte vybraný problém z oblasti prepojených vozidiel a na základe analýzy navrhnite simulačný nástroj alebo navrhnite a implementuje metódu overenia riešenia v reálnom prostredí. Overenie vykonajte prostredníctvom simulácie, alebo reálneho merania. Výsledky zhodnoťte a porovnajte s inými publikovanými prácami.

*zadná strana zadania/revízie (môže byť prázdna)*

Čestne vyhlasujem, že som túto prácu vypracoval(a) samostatne, na základe konzultácií a s použitím uvedenej literatúry.

V Bratislave, 9.5.2022

Jakub Ganádik

*zadná strana čestného prehlásenia (prázdna strana)*

**Anotácia**

Slovenská technická univerzita v Bratislave

FAKULTA INFORMATIKY A INFORMAČNÝCH TECHNOLÓGIÍ

Študijný program: Informatika

|  |  |
| --- | --- |
| Autor: | Jakub Ganádik |
| Bakalárska práca: | Analýza možností výmeny informácií v prostredí navzájom prepojených vozidiel |
| Vedúci bakalárskej práce: | Lukáš Šoltés |
| Máj 2022 |  |

V posledných rokoch sa čoraz častejšie skloňujú pojmy ako “smart device” alebo “internet of things”, výnimkou nie je ani oblasť automobilovej dopravy. S rastúcim počtom vozidiel na cestách a vyšším dôrazom na bezpečnosť bude v budúcnosti dôležité aby vozidlá patrili do navzájom prepojených sietí a vedeli komunikovať so svojím okolím a sebou navzájom. Predtým ako toto bude možné treba urobiť ešte veľa testov, keďže testovanie v reálnych podmienkach je zdĺhavé a drahé táto práca sa zameriava na zistenie možností testovania takýchto scenárov pomocou už existujúcich softvérových riešení. Práca sa venuje výberu vhodného dopravného a sieťového simulátoru, tak aby potom tieto simulátory vedeli medzi sebou spolupracovať a vytvoriť simuláciu skutočného miesta s premávkou vozidiel a aby tieto vozidlá boli schopné medzi sebou komunikovať. Následne sa v tejto spojenej simulácii testujú schopnosti a obmedzenia použitých simulátorov, rôzne nastavenia aby bola čo najbližšie skutočným scenárom. Nakoniec sa ešte v práci testuje možnosť komunikácie medzi simuláciou a reálnym svetom s použitím skutočných paketov.

*zadná strana anotácie (prázdna strana)*

**Annotation**

Slovak university of Technology Bratislava

FACULTY OF INFORMATICS AND INFORMATION TECHNOLOGIES

Degree course: Informatika

|  |  |
| --- | --- |
| Author: | Jakub Ganádik |
| Bachelor’s Thesis: | Analýza možností výmeny informácií v prostredí navzájom prepojených vozidiel |
| Supervisor: | Lukáš Šoltés |
| 2022 may |  |

In recent years, terms such as "smart device" or "internet of things" have become more and more common, with the car industry being no exception. With the growing number of vehicles on our roads and a greater emphasis on safety, it will be important in the future for vehicles to be interconnected in networks and to be able to communicate with their surroundings and with each other. Many tests still need to be done before that can happen. Real-world testing is expensive and difficult that is why this work focuses on identifying the possibilities of testing such scenarios using existing software solutions. The work is devoted to the selection of a suitable traffic and network simulator, so that these simulators can then cooperate with each other and create a simulation of a real place with vehicle traffic and so that these vehicles are able to communicate with each other. Subsequently, in this combined simulation, the capabilities and limitations of the simulators used are tested, including tests of various settings in order to make the simulation as real-life as possible. Finally, the possibility of communication between the simulation and the real world using real packets is tested in the work.

*zadná strana anotácie (prázdna strana)*

# Obsah

[Úvod 5](#_Toc101431849)

[Analýza 8](#_Toc101431850)

[Simulátor dopravy 8](#_Toc101431851)

[Sieťový simulátor 8](#_Toc101431852)

[SUMO 9](#_Toc101431853)

[Použité nástroje v SUMO 9](#_Toc101431854)

[SUMO v porovnaní s inými simulátormi dopravy 13](#_Toc101431855)

[ns-3 13](#_Toc101431856)

[Príklady využitia ns-3 a SUMO zo skutočného života 15](#_Toc101431857)

[Windows subsystem for Linux 16](#_Toc101431858)

[Špecifikácia požiadaviek **Error! Bookmark not defined.**](#_Toc101431859)

[Cieľ 18](#_Toc101431860)

[Funkčné požiadavky 18](#_Toc101431861)

[Nie-funkčné požiadavky 18](#_Toc101431862)

[Návrh riešenia 19](#_Toc101431863)

[Testovacie scenáre **Error! Bookmark not defined.**](#_Toc101431864)

[Práca do budúcnosti 31](#_Toc101431865)

[Literatúra 33](#_Toc101431866)

[Príloha – plán práce na projekte 35](#_Toc101431867)

## 

## Úvod

Množstvo vozidiel na našich cestách sa každým rokom zvyšuje a s ním aj nevyhnutnosť na použitie systému, ktorý by zabezpečil komunikáciu medzi vozidlami (V2V), vozidlami a infraštruktúrou (V2I), vozidlami a cestou (V2R), vozidlami a všetkým (V2X). Systémy na medzivozidlovú komunikáciu sú počítačové systémy v ktorých vozidlá predstavujú sieťové zariadenia. Vozidlá sú schopné komunikovať medzi sebou a inými sieťovými zariadeniami umiestnenými popri ceste. Zariadenia si vymieňajú rôznorodé informácie, hlavne bezpečnostné upozornenia a dopravné informácie. Tieto informácie môžu slúžiť na vyhýbanie sa dopravným nehodám alebo zmierňovanie kolón. Komunikácia medzi vozidlami je zvyčajne vyvíjaná ako súčasť inteligentných dopravných systémov. Začiatky ich vývoja sa datujú späť do roku 70. rokov minulého storočia. Na prenos dát boli používané rôzné prenosové médiá napr. lasery, rádiové vlny a iné. V súčasnosti najpoužívanejší spôsob komunikácie medzi vozidlami je tzv. Vehicular Ad Hoc Network (VANET). Vanet funguje na princípe spontánneho vytvárania bezdrôtových systémov mobilných zariadení (vozidiel). VANET-y boli prvý krát predstavené v roku 2001. Majú rôzne možnosti využitia napríklad zabraňovanie kolíziám, dynamické plánovanie trás, sledovanie stavu premávky v reálnom čase atď. Z týchto dôvodov sa očakáva, že VANET-y budú bežne využívané na vozidlách budúcnosti.

Predtým ako budú VANET-y implementované v bežnej premávke je potrebné vykonať realistické počítačové simulácie VANET-ov s použitím simulátorov dopravy a sieťových simulátorov. Simulácia VANET-ov vyžaduje použitie dvoch komponentov: simulátor dopravy, ktorý by poskytol presný model pohybu pre node-y vo VANET-e a sieťový simulátor, ktorý by simuloval bezdrôtovú sieť. [1] [4] [5] [6]

V tejto práci sa budem sústrediť na vytvorenie simulácie dopravy pomocou simulátora SUMO v kombinácii so sieťovým simulátorom ns-3 a budem zisťovať, či dokážu simulovať komplexné dopravné situácie z reálneho života so zameraním na komunikáciu medzi vozidlami. Bude sa tiež možnosť komunikácie reálnych zariadení so simuláciou.

## Analýza

### Simulátor dopravy

Simulácia dopravy je matematické modelovanie dopravných systémov, takýmto spôsobom sa modelujú napríklad zložité diaľničné križovatky, dôležité dopravné uzly, ulice, križovatky atď. Vďaka simulátoru je možné jednoduchšie navrhovať a plánovať skutočné dopravné systémy. Vývoj simulátorov dopravy začal pred cca 40 rokmi. Veľa národných dopravných agentúr a akademickýchh inštitúcii ich používa pri simulovaní dopravných situácii. Simulácia v doprave dokáže sledovať modely príliš komplikované na numerické spracovanie, môže sa použiť na experimentálne štúdie a pomáha pri vytváraní komplikovaných scenárov v doprave.

Simulácie dopravy spadajú do nasledovných 2 kategórii:

1. Makroskopická simulácia: Sledujeme dopravnú situáciu ako celok, dopravný tok.

2. Mikroskopická simulácia: Zameriava sa na individuálne vozidlá a interakciu medzi nimi. Napríklad zisťovanie rýchlosti vozidiel, ich akceleráciu, správanie sa vodiča... [5]

### Sieťový simulátor

Sieťový simulátor je softvér, ktorý sa snaží replikovať správanie skutočnej siete. To je docielené zisťovaním interakcií medzi rôznymi sieťovými zariadeniami napr. prepínačmi, smerovačmi, AP bodmi... Simulátory zväčša simulujú udalosti, stavové premenné sa menia v diskrétnych časových momentoch. Správanie siete a rôznych aplikácií a služieb je možné sledovať v testovacom prostredí, ktorého atribúty sa môžu upravovať , vďaka tomu je možné na danej sieti otestovať v rôznych podmienkach rôzne scenáre aj bez toho aby musela daná sieť fyzicky existovať. Testovanie siete takýmto spôsobom šetrí čas aj peniaze. Simulátory, ako napríklad ns-3, ktoré budem používať, sa používajú na simuláciu sieťových a smerovacích protokolov. [12] [7]

#### Sieťová emulácia

Emulátor siete sa používa na testovanie výkonu skutočnej siete. Emulátory sú vhodné aj na troubleshooting skutočných sietí. Sieťový emulátor môže byť dostupný ako hardvérové ​​alebo softvérové ​​riešenie. V ns-3 je možné posielať skutočné pakety zo živej aplikácie cez simuláciu virtuálnej siete. Skutočný paket sa „moduluje“ do simulačného paketu. [7]

### SUMO

"Simulation of Urban Mobility” (SUMO) je mikroskopický a open source simulátor. Umožnuje simuláciu dopravy, ktorá sa skladá zo samostatných vozidiel, ktoré sa pohybujú po predtým zadanej cestnej sieti. SUMO umožňuje simulovať veľké množstvo rôznych dopravných scenárov. Simulácia je výlučne mikroskopická teda každé vozidlo je presne definované menom, časom odjazdu zo štartovacej hrany a štartovacou a koncovou hranou. Vozidlá môžu byť upravené aj detailnejšie napr. je možné meniť rýchlosť vozidla. Každému vozidlu je pridelený typ, ktorý hovorí o fyzických vlastnostiach vozidla. Simulácie sú deterministické, ale existujú riešenia na implementáciu náhodnej dopravy.

SUMO sa často používa na výskum výberu trás pre autá, algoritmov na spúšťanie semafórov alebo na skúmanie komunikácie medzi vozidlami. Je používaný na simulovanie rôznych projektov, ktoré zahŕňajú autonómne vozidlá alebo stratégie na riadenie dopravy.

Pravdepodobne najpopulárnejšie použitie SUMO je modelovanie dopravy na výskum V2X komunikáciu. [8]

### Použité nástroje v SUMO

#### Sumo

Samotná simulácia, ktorá je kontinuálna v priestore a diskrétna v čase. Je to mikroskopická simulácia pohybu dopravy. Jeho úlohou je simulovať zadefinované scenáre. Je spustiteľný na Linux alebo Windows. Vstupom pre Sumo je vygenerovaná cestná sieť, buď cez netgenerate, netconvert... V mojom prípade sa na generovanie cestnej siete používa OSM Web Wizard. Vstupom môžu byť tiež vygenerované cesty napr. z jtrrouter alebo duarouter. Do Sumo môžeme vkladať aj prídavné semafóry, cestné značenie... Tieto sú celkom vhodné keďže Sumo má niekedy problém určiť správnu pozíciu semafórov na mapách vygenerovaných OSM Web Wizard. Vizualizácia cestnej siete sa uskutočňuje pomocou sumo-gui. Pomocou Sumo môžeme získať výstup vo vela rôznych formátoch. Ja budem používať “.xml” format. [8]

#### OSM Web Wizard

OSM Web Wizard ponúka jedno z najjednoduchších a najviac priamočiarych riešení, ako začať so SUMO. Vďaka OSM Web Wizard je možné vybrať časť mapy z openstreetmap a následne na nej nakonfigurovať náhodné požiadavky na premávku a spustiť a vizualizovať scenár v sumo-gui.

Po spustení je možné vybrať si akékoľvek miesto na svete. Zaškrtnutím Select Area sa začína výber požadovanej oblasti. Kliknutím na tlačítko “Generate Scenario” sa vygeneruje scenár t.j. už spustiteľná simulácia v sumo-gui. Simulácia obsahuje všetky ulice, ktoré sa nachádzajú vo svetlej časti obrazovky počas kliknutia na generovanie scenáru.

OSM Web Wizard umožňuje ponastavovať niektoré parametre simulácie ešte pred jej vygenerovaním. SUMO totiž podporuje rôzne spôsoby dopravy. Na paneli generovania dopytu sa dajú aktivovať/deaktivovať jednotlivé spôsoby dopravy. Pre každý spôsob dopravy OSM Web Wizard generuje náhodný dopyt, tento dopyt je ovplyvnený dvomi parametrami:

Through Traffic Factor – Zakaždým, keď sa vygeneruje nové vozidlo, OSM Web Wizard náhodne vyberie štartovaciu a koncovú hranu pre vozidlo. Faktor definuje, pravdepodobnosť s ktorou sa vyberú hrany vo vnútri simulácie narozdiel od hrán na okraji.

Count – Množstvo vygenerovaných vozidiel za hodinu a kilometer cesty resp. jazdného pruhu.

Map

Description automatically generated

OSM Web Wizard ukladá celý simulačný scenár sumo config a prechodné súbory do lokálneho adresára, ktorý je pomenovaný časom vytvorenia simulácie teda napr. „2016-10-17-14-54-30”. Tu sa nachádzajú “.xml” súbory, ktoré slúžia na ďaľšiu modifikáciu danej simulácie. Môžem tu napríklad upraviť trasy vozidiel alebo dokonca úplne zrušiť náhodné cesty pre vozidlá. Dajú sa tu napríklad pridávať aj nové cesty a takto vytvárať úplne imaginárne scenáre. V neposlednom rade sa tu dajú upravovať už existujúce cesty – dá sa meniť typ ciest napr. z cesty pre autá spravím cestu pre bicykle( zíde sa obzvlášť v prípadoch kedy sú vo vygenerovanej simulácii niektoré cesty chybne označené). [9]

#### TraceExporter

traceExporter.py konvertuje a filtruje výstup SUMO fcd do rôznych formátov „trace files“ teda súborov obsahujúcich pohyby vozidiel v simulácii. V tejto simulácii sa výstup SUMO fcd bude konvertovať do súboru s príponou “.tcl - Tool Command Language”, tento typ súboru je používaný simulátormi ns2/ns3. SUMO fcd výstup (floating car data) obsahuje rýchlosť, polohu a ďaľšie dôležité informácie pre každé vozidlo v každom momente. Výstup by sa dal interpretovať ako super presné vysokofrekvenčné GPS zariadenie pre každé vozidlo. [10]

#### Sumo-gui

V postate to isté ako SUMO, ale s grafickým rozhraním. Je to vizualizácia cestnej siete. Vieme v ňom otvoriť konfiguračný súbor SUMO. Samotná vizualizácia je potom jednoducho nastaviteľná. Môžeme určiť výpis časových krokov v sekundách/minútach/hodinách. Delay nám určuje rýchlosť simulácie, doslova to je oneskorenie v ms medzi jednotlivými krokmi simulácie. Na podrobnejšie sledovanie slúžia aj breakpointy, ktoré môžeme umiestniť do jednotlivých časov simulácie. Scale traffic umožňuje zvýšiť množstvo vozidiel v simulácii. S pomocou GUI vieme zistiť, ktoré cesty sú vhodné pre dané typy vozidiel, sivé cesty napríklad neumožňujú premávku osobných áut, čierne na druhú stranu áno, Tieto farby je dôležité sledovať, pretože niekedy sa stáva, že cesty, ktoré sú v skutočnosti vhodné pre autá sú v SUMO označené ako nevhodné, preto tieto cesty treba upraviť v súbore “osm.net.xml”, ktorý sa nachádza v predtým vygenerovanom adresári “2016-10-17-14-54-30”. V prípade, že chceme používať manuálne generované vozidlá, sumo-gui umožňuje skopírovať id jazdného pruhu, toto id potom môžeme použiť ako štartovaciu/konečnú hranu pre vozidlo. V neposlednom rade je možné interagovať s takmer všetkými objektami v simulácii, pravým klikom môžeme napríklad meniť nastavenia semafórov, sledovať vozidlá, meniť nastavenia cestného značenia, jazdných pruhov, križovatiek. Nachádza sa tu aj menu na zmenu farieb vďaka nemu môžeme napríklad meniť veľkosti a farbu vozidiel.

Graphical user interface, map

Description automatically generated

Na obrázku vyššie je vidieť spomínaný problém s cestami. Normálne cesty okolo fakulty sú označené ako nevhodné pre autá. [11]

#### Netedit

Je vizuálny sieťový editor. V Netedit vieme pridávať nové cesty, križovatky, semafóry… Môže byť použitý na vytváranie testovacích scenárov s úplne imaginárnou cestnou sieťou alebo pomocou neho môžeme upraviť už existujúce scenáre. Vďaka výkonnému rozhraniu výberu a zvýrazňovania ho možno použiť aj na ladenie sieťových atribútov. Používateľské rozhranie úzko kopíruje rozhranie sumo-gui.

**Chart

Description automatically generated**

Ukážka Netedit, tvorenia edges a rôznych módov, ktoré sat u nachádzajú. [12]

### SUMO v porovnaní s inými simulátormi dopravy

Existuje veľa iných pomerne populárnych simulátorov dopravy. Napríklad: Aimsun, Paramics Microsimulation, Treiber’s Microsimulation of Road Traffic... SUMO má v porovnaní s ostatnými spomínanými simulátormi hneď niekoľko výhod. Je síce Pravda, že Paramics a Aimsun sú 3D simulátory narozdiel od 2D SUMO a Treiber’s Microsimulation of Road Traffic a zdá sa, že samotné scenáre sú ľahšie upraviteľné v samotnom GUI. Avšak vďaka NetEdit sa dajú upravovať aj scenáre v SUMO a 2D vs 3D simulácia tiež nie je problém, keďže nakoniec nás aj tak zaujíma len pohyb vozidiel. Dôležitejší rozdiel medzi jednotlivými simulátormi je v cene. Paramics Microsimulation, ponúka po prihlásení len free trial, potom je spoplatnený. Aimsun taktiež ponúka len 30 dňový free trial. Z týchto štyroch simulátorov som nakoniec zvažoval len SUMO a Treiber’s Microsimulation of Road Traffic (TMORF), ktoré sú open source a free to use. Aj medzi nimi existujú rozdiely, kým SUMO je vyvíjané dvomi inštitúciami, TMORF je osobný projekt jedného človeka. TMORF nanešťastie neobsahuje GUI na úpravu dopravnej siete, ale hlavne nie je schopné načítať reálny scenár zo skutočného miesta. [3]

Kvôli týmto dôvodom som sa rozhodol použiť SUMO ako môj simulátor dopravy.

### ns-3

Je jeden zo skupiny sieťových simulátorov, ktoré sú využívané predovšetkým na výskum a vyučovanie. Jeho používanie nie je spoplatnené. Obsahuje pomerne rozsiahlu dokumentáciu a veľké množstvo tutoriálov. Softvérová štruktúra ns-3 z neho robí vhodný nástroj na realtime emuláciu, teda pri prepojení s vonkajším svetom. Ns-3 je hlavne využívané na bezdrôtové IP simulácie, ktoré zahŕňajú modely pre wifi, wifimax... Poskytuje tiež množstvo statických a dynamických protokolov napr. AODV, OLSR atď. Ns-3 je možné používať ako emulátor, teda ns-3 dokáže generovať pakety pre reálne zariadenia a tiež môže slúžiť ako linka medzi virtuálnymi strojmi. Ns-3 funguje Linux MacOS a Windows, ale niektoré funkcionality ako napríklad emulátor nie sú dostupné na Win. [14] [15]

#### Stručný popis simulačných scenárov

Ns-3 nemá pekné grafické používateľské rozhranie a všetky simulačné scenáre sú písané v C++ alebo pythone s použitím simulačného jadra ns-3. Po pridaní potrebných header files nám ns-3 umožňuje upravovať scenáre rôznymi spôsobmi. V prvom rade sa vytvárajú node objekty tieto samotné nerobia z pohľadu simulácie nič, môžeme ich ale neskôr medzi sebou prepojiť pomocou vytvorenej topológie. Následne ns-3 pomáha s vytváraním abstrakcii nad reálnym svetom. Medzi danými nodes potrebujeme urobiť linku a na to použijeme tzv. Helper (PointToPoint, Csma…), ktorá prepojí pomocou Channelu naše NetDevices (tieto sú tiež abstrakcie, v skutočnosti sú to sieťové karty a sieťové káble). Atribúty kanálov a NetDevices sú tiež nastaviteľné ak používame wifi channel tak môžeme vybrať, či je daný node STA/AP/Ad hoc. Potom sa používa tiež InternetStackHelper, ktorý nainštaluje Internet stack teda IP, UDP, TCP … na každý node v predtým určenom kontajnery. Jednotlivým zariadeniam potom môžeme pomocou Ipv4AddressHelper priradiť ip adresu z určeného rozsahu. Nakoniec môžeme z vybraných nodes vytvoriť UDP echo server a client alebo môžeme ich komunikáciu zabezpečiť pomocou socketov alebo si môžeme medzi nimi posielať ručne vygenerované pakety… [16] [17]

Na vizualizáciu daného scenáru používam NetAnim a Wireshark.

#### NetAnim

Je to offline animačný nástroj, ktorý je v súčasnosti schopný animovať simuláciu s použitím “.xml” súboru zozbieraného počas simulácie. Dokáže simulovať prenos paketov po bezdrôtových aj drôtových linkách. Zobrazuje nodes v simulácii s ich pozíciou a pohybmi (dokáže vypísať aj trajektóriu pohybu pre daný node), taktiež o každom node poskytuje základné informácie (ip, mac adresu, súradnice). Ďalej zobrazuje prúd paketov medzi vybranými nodes. Samozrejme môžeme simuláciu spomaliť/zrýchliť alebo zastaviť. Do pozadia je možné umiestňovať obrázky (vhodné pri kombinácii s fotografiou zo sumo-gui).

Calendar

Description automatically generated with medium confidence

Červené body na obrázku predstavujú AP body pozdĺž diaľnice a body, ktoré sa nachádzajú na vrchu obrázku s prekrývajúcimi sa id sú vozidlá. Na začiatku simulácie sú všetky vozidlá na svojej východiskovej hrane aj tie, ktoré ešte nevyrazili a teda ich nebolo vidieť v SUMO simulácii. Z tohto dôvodu robia náhodne generované scenáre nečitateľnú spleť bodov. [18]

NS-3 sa primárne používa na simuláciu, ale je tiež schopný spúšťať emulácie na paketoch v reálnom živote. Niektoré aspekty ns-3 závisia od podpory Unixu (alebo konkrétne Linuxu), ako je napríklad emulácia. Z tohto dôvodu som sa rozhodol nainštalovať ns-3 na Windows Subsystem for Linux, pre prípad, že by niekto chcel v budúcnosti spúšťať emulácie. [17]

#### ns-3 v porovnaní s inými sieťovými simulátormi

Porovnával som hlavne známejšie sieťové simulátory, ktoré sa celkom rozšírene používajú v kombinácii so SUMO. Teda ns3, OMNeT++ a NetSim.

NetSim vyzerá byť najviac používaný v prostredí súkromných firiem a narozdiel od ns-3 prichádza s už vstavaným interfaceom pre SUMO, čo robí vizualizácie mestského prostredia oveľa zrozumiteľnejšie a prehľadnejšie ako v ns-3 (v ns-3 sa robí vizualizácia pomocou obrázku na pozadí). Nanešťastie je jediný z týchto troch simulátorov, ktorý je platený a preto je nevhodný.

OMNeT++ a ns-3 sa obidve zdajú byť v istom zmysle veľmi podobné, ale sú medzi nimi určité rozdiely. Najdôležitejšie rozdiely budú GUI, OMNeT ho má prepracovanejšie a oveľa viac funkčné ale dokumentácia sa zdá, že je lepšia pri ns-3. Nakoniec som sa práve kvôli veľkému množstvu tutoriálov, dokumentácie a videí rozhodol použiť ns-3. [19] [20]

### Príklady využitia ns-3 a SUMO zo skutočného života

iTETRIS – V poslednej dobe neustále rastie záujem o V2X komunikáciu (komunikácia vehicle to everything), nanešťastie reálne nasadenie tejto technológie je nákladné a použitie ad-hoc systému na ovládanie premávky je potencionálne nebezpečné. Pre účely výskumu bol potrebný framework, ktorý by simuloval interakciu medzi vozidlami a infraštruktúrov. Cieľom iTeTRIS projektu bolo vyrobiť takýto framework spojením ns-3 a SUMO pomocou iTETRIS Control System. Práca v rámci iTETRIS obsahovala veľké množstvo príprav. Tie sa sústreďovali najmä na zisťovanie problémov skutočnej premávky a ich modelovanie v simulačnom prostredí. Projekt bol spolufinancovaný mestom Bologna, ktoré podporovalo dopravnú simuláciu pokrývajúcu rôzne časti mesta. “One of the project’s outputs is a set of in-depth descriptions of V2X-based traffic management applications, including different attempts for traffic surveillance, navigation, and traffic light control. In the following, one of these applications, the bus lane management, is described, showing the complete application design process, starting at problem recognition, moving over the design of a management application that tries to solve it, and ending at its evaluation using the simulation system.” [1]

Podobne ako v iných európskych mestách aj v Bologni je verejná doprava veľmi dôležitá. Preto sa tu nachádza množstvo BUS pruhov a ulíc vyhradených čisto pre verejnú dopravu. Mesto, ale čelý veľkým verejným podujatiam (futbalové zápasy, výstavy...) a počas nich nastáva k zvýšenej premávke. Jeden z nápadov v iTETRIS zahŕňa dočasné otváranie BUS pruhov pre individuálnu dopravu. Aplikácia mala obsahovať dva podsystémy. Jeden z nich bol zodpovedný za zisťovanie stavu premávky na cestách. Druhý zisťoval, či sa oplatí otvoriť BUS pruhy pre autá a zároveň posielal túto informáciu zapojeným vozidlám. Stav dopravy bol zisťovaný pomocou Cooperative Awareness Messages a Road Side Units. Pri znížení priemernej rýchlosti vozidiel sa predpokladá zahustenie premávky, táto metóda však nie je rozšírená a preto vyžaduje ďaľšie štúdium. Nakoniec sa simuláciou nepodarilo zistiť výrazný rozdiel medzi premávkou pred a po otvorení BUS pruhov. Hlavne kvôli tomu, že autá často stoja za autobusmi a naopak spomaľujú autobusy. [1]

### Windows subsystem for Linux

#### WSL

Windows Subsystem for Linux je vrstva kompatibility pre spustenie Linuxových binárnych súborov natívne v prostredí konzoly v systéme Windows 10. Nevyžaduje sa žiadna opätovná kompilácia ani portovanie. WSL poskytuje rozhranie jadra kompatibilné s Linuxom vyvinuté spoločnosťou Microsoft a umožňuje používateľovi vybrať si distribúciu Linuxu na inštaláciu z obchodu Microsoft Store. Distribúcia Linuxu poskytuje správu binárnych balíkov v kontajnerovom prostredí. WSL poskytuje rozhranie na pripojenie jednotiek v rámci WSL, napríklad c:\ sa automaticky pripojí ako /mnt/c.

Windows Subsystem for Linux umožňuje spúšťať prostredie Linux priamo v systéme Windows, bez úprav, bez réžie tradičného virtuálneho stroja a bez nutnosti duálneho bootovania. To zahŕňa aj príkazový riadok a jeho nástroje, pomôcky, aplikácie. WSL funguje na Windows 10, Windows 11 a Windows Server 2019. [21] [22]

#### Dôvody pre WSL

NS-3 je primárne vyvinutý, podporovaný a testovaný na platformách Linux a macOS. Dokumentácia pre ns-3 hovorí:

“The project tries to support most or all of the build options on these platforms unless there is a good reason to exclude the option; and at least the debug build will compile. If you intend to do serious work using NS-3 and are forced by circumstances to use a Windows platform, consider virtualization of a popular Linux platform or using Windows Subsystem for Linux “. [17]

#### WSL vs virtual machine

WSL vyžaduje menej zdrojov (CPU, pamäť a úložisko) ako virtuálny stroj. WSL mi tiež umožňuje spúšťať nástroje a aplikácie príkazového riadka systému Linux a zároveň umožňuje používať pracovnú plochu a príkazy Windows. Tiež je možné pristupovať k súborom systému Windows z WSL.

Na druhej strane, ak by som niekedy potreboval prístup ku grafickým aplikáciám na Linuxe, virtuálny stroj sa zdá byť lepšou voľbou, aj keď Microsoft sľubuje podporu pre GUI aplikácie vo WSL. Zatiaľ to je možné len s použitím ďaľšieho softvéru.

Ak by som chcel vyskúšať rôzne distribúcie, bolo by mi lepšie používať VM. Aj keď WSL ponúka výber distribúcií, výber v Microsoft Store je stále obmedzený.

Nakoniec som sa rozhodol použiť wsl. Aj keď veľa distribúcií nie je dostupných v obchode Microsoft Store, rozhodol som sa, že akékoľvek rozumné distribúcie budú vhodné pre moje súčasné potreby a potreby bakalárskej práce. Budem používať ubuntu 20.04. Jediným skutočným problémom bolo GUI pre WSL. Všetky simulátory, ktoré som chcel použiť, majú GUI. Aplikácie GUI nie sú podporované, ale našiel som niekoľko riešení, ktoré boli dosť jednoduché. Bola to jednoduchosť a rýchlosť nastavenia, čo ma prinútilo vybrať si wsl namiesto virtual machine. [23]

#### Xming

Je to display server pre operačné systémy Windows. Server spracuje požiadavky a zobrazí ich pre používateľa ako grafický výstup. [24]

Vďaka Xming som bol schopný použiť WSL a zároveň používať grafické rozhranie v používaných aplikáciách.

## Opis riešenia

### **Špecifikácia požiadaviek**

### Cieľ

Nájsť a otestovať simulačné prostredie/prostredia vďaka ktorým by som mohol simulovať komunikáciu medzi vozidlami a okolím. Simuláčný scenár by mal zodpovedať skutočnému miestu a skutočným vozidlám. Simulačné scenáre vo všetkých simulátoroch by mali byť vysoko modifikovateľné. Simulátory by mali spĺňať aj emuláciu, teda na testovanie sieťového simulátoru by malo byť možné použiť reálnu sieťovú premávku. Všetky potrebné simulátory a ich súčasti by mali fungovať na mojom stroji.

### Funkčné požiadavky

1. Všetky simulácie by mali byť spustiteľné na mojom zariadení.
2. WSL by malo byť shopné spustiť GUI jednotlivých simulátorov a použitého softvéru
3. Simulovanie reálnej lokality. Mal by som byť schopný si vybrať akékoľvek miesto na svete (alebo aspoň v Bratislavskom kraji) a simulovať tam cesty, budovy, cestné značenie, jazdné pruhy…
4. Simulačný scenár by mal byť jednoducho upraviteľný. Napríklad by som mal byť schopný odstrániť a zmeniť cestné značenie, semafóry…
5. Testovacia simulácia. Vytvorenie úplne nového imaginárneho prostredia. Čiže vytvorenie novej mapy od základu vrátane ciest, križovatiek…
6. Simulácia vozidiel a ich trajektórii. Toto zahŕňa manuálne generovanie vozidiel a ich trás a tiež v prípade väčších simulácii automatické náhodné generovanie vozidiel.
7. Pri automatickom generovaní vozidiel by som mal byť schopný upravovať objem dopravy (množstvo vozidiel).
8. Simulovanie pohyblivých sieťových zariadení, vozidiel, ktoré by medzi sebou komunikovali.
9. Simulovanie stacionárnych sieťových zariadení. Mal by som byť schopný umiestniť AP body na mnou vybrané miesta. Tieto by mali zabezpečovať komunikáciu medzi vozidlami a zvyškom simulácie/emuácie.
10. Sieťový simulátor by mal mať možnosť upraviť simuláčný scenár, ktorý prišiel z dopravného simulátoru. Malo by byť možné pridať budovy, cesty, vozidlá. Vozidlám zmeniť rýchlosť, smer...
11. Sieťová simulácia by mala byť schopná pracovať s akoukoľvek dopravnou simuláciou vytvorenou v dopravnom simulátore
12. Dĺžka sieťovej simulácie by mala byť nastaviteľná a nezávislá od dĺžky dopravnej simulácie
13. Jednotlivé uzly (nodes) v simulácii by mali byť schopné používať rôzne druhy fyzickej vrstvy z OSI modelu
14. Vlastnosti komunikačných kanálov medzi jednotlivými zariadeniami by mali byť nastaviteľné
15. Vlastnosti AP bodov (vysielačov) by mali byť nastaviteľné
16. Pri použití 802.11 by mali mať všetky uzly možnosť prepínať medzi rôznymi módmi wifi.
17. Uzly by mali byť schopné medzi sebou komunikovať rôznymi protokolmi transportnej vrstvy OSI
18. Mal by sa dať nastaviť uzol, ktorý bude schopný generovať pakety
19. Mal by sa dať nastaviť uzol, ktorý bude prijímať vygenerované pakety
20. Všetky uzly by mali mať nastaviteľnú IP adresu a masku.
21. Sieťový simulátor by mal podporovať rôzne druhy routovacích protokolov.
22. Spôsob routovania by mal byť nastaviteľný.
23. Celú simuláciu by malo byť možné sledovať na GUI.
24. Animovanie komunikácie medzi danými zariadeniami. Malo by byť vidieť sieťovú premávku medzi jednotlivými zariadeniami.
25. Mal by som byť schopný zachytávať sieťovú premávku na vybraných zariadeniach a ich rozhraniach.
26. Moje zariadenie by malo byť schopné posielať pakety zariadeniam v simulácii
27. Zariadenia v simulácii by mali byť schopné posielať pakety von na rozhranie môjho zariadenia.
28. Sieťový simulátor by mal byť schopný komunikovať s niekoľkými reálnymi rozhraniami naraz.
29. Sieťový simulátor by mal vedieť si vytvoriť vlastné rozhranie na skutočnom zariadení
30. Sieťový simulátor by mal vedieť po konci simulácie vytvorené rozhranie zničiť.
31. V prípade komunikácie s reálnym svetom by táto komunikácia mala prebiehať v reálnom čase.

### Nie-funkčné požiadavky

1. Sieťová premávku by malo byť možné zachytiť pomocou network protocol analyzeru najlepšie by bolo použiť wireshark.
2. Semafóry, cesty, budovy by mali byť pre jednotlivé scenáre upraviteľné v textových súboroch.
3. Manuálne generovanie vozidiel by malo byť možné úpravou jedného súboru.
4. Simulácia by mala vedieť simulovať komunikáciu uzlov pomocou wifi, ethernetu a point-to-point.
5. Uzly v sieťovej simulácii by mali byť schopné posielať a rozumieť UDP/TCP/ICMP/ARP paketom
6. Simulácia by mala mať nastaviteľný propagation delay, pre jednotlivé komunikačné kanály.
7. Ap body a vozidlá by mali byť schopné spolu komunikovať len na určitú vzdialenosť. S použitím logarithmic propagation loss model a s nastaviteľnou referenčnou vzdialenosťou podľa zaužívaných štandardov v mestskom prostredí. “For outdoor propagation, the reference distance d0 of 10 m or 100 m is recommended.” [2]
8. Uzly by mali byť schopné medzi sebou komunikovať s constant propagation loss, teda napríklad bez prekážok a nezávisle na vzdialenosti.
9. AP by malo byť schopné posielať beacon frames každému zariadeniu so station módom.
10. Mal by som byť schopný meniť interval posielania beacon frames
11. Jednotlivé uzly by mali byť schopné fungovať v módoch access point, station a ad-hoc. Ideálne naraz v rôznych podsieťach.
12. Uzol generujúci pakety by mal byť schopný poslať pakety na určenú IP adresu a port.
13. Simulátor by mal podporovať globálne routovanie pre menšie simulácie.
14. Simulátor by mal podporovať OLSR routovací protokol.
15. Uzly používajúce OLSR by mali generovať “hello” a “topology control” správy.
16. Všetky simulácie by sa mali dať spustiť na x64 Windows 10.
17. Simulácie by mali bežať na windows subsystem for linux 1 alebo 2
18. WSL by malo používať display server na spustenie GUI
19. Vozidlá by mali byť schopné medzi sebou komunikovať v režime ad hoc. A presúvať si pakety smerujúce k najbližšiemu AP bodu medzi sebou.
20. Wireshark by mal byť schopný zachytiť pakety smerujúce zo simulácie na rozhraní môjho zariadenia
21. Mal by som byť schopný pingnúť zariadenia v simulácii zo svojho zariadenia
22. Simulácia by mala vytvoriť reálne rozhranie na základe informácií zadaných v simulácii.

### **Návrh riešenia**

1. Úplne ako prvú vec budem musieť vyriešiť problém spustiteľnosti na mojom zariadení. Vzhľadom na to, že väčšina ns-3 tutorálov je na Linux a ns-3 dokumentácia hovorí toto: “If you intend to do serious work using ns-3, and are forced by circumstances to use a Windows platform, consider virtualization of a popular Linux platform or using Windows Subsystem for Linux” Takže sa bude používať buď WSL alebo virtual machine. Na základe predošlých informácii bude použité WSL.
2. V prípade použitia WSL bude treba doinštalovávať nejaký display server napr. XLaunch , keďže bez neho by sa nedali spustiť grafické rozhrania simulátorov SUMO, ns-3 a nedal by sa použiť ani wireshark.
3. Inštalácia dopravného simulátora Simulator of urban mobility SUMO.
4. Inštalácia sieťového simulátora ns-3.
5. Výber miesta na ktorom sa bude testovať. Na otesovanie schopností SUMO, do úvahy pripadá na testovacie účely Račianske mýto, keďže je to zložitá a frekventovaná križovatka.
6. Vytvorenie scenáru z vybratého miesta z open maps. Na to sa použije OSM Web Wizard.
7. Následne bude treba upraviť daný vygenerovaný scenár. Môže sa napríklad stať, že semafóry nebudú na správnych miestach alebo niektoré cesty budú označené ako nevhodné pre vozidlá. Toto sa bude upravovať vo vygenerovaných xml súboroch.
8. Na otestovanie ns-3 v kombinácii so SUMO by sa dalo použiť okolie fakulty, nie je tu veľa ciest a preto sa tu dajú otestovať základné funkcionality.
9. Po úprave samotnej SUMO simulácie nasleduje úprava trás vozidiel. V testovacích scenároch budem generovať vozidlá manuálne, budem upravovať ich štartovací a koncový jazdný pruh(hranu) a čas odjazdu (spawnu).
10. Vytvorenie „.xml“ súboru zo súradníc pohybu vozidiel.
11. Úprava predtým vygenerovaného „.xml“ súboru do formátu, ktorému rozumie simulátor ns-3, teda „.tcl“.
12. Nakoniec by som ostatné funkcionality testoval na dlhom rovnom úseku diaľnice aby tu bolo možné sledovať komunikáciu medzi zariadeniami. To zahŕňa všetky predošlé kroky a nasledujúce.
13. Výber súradníc v sumo-gui pre umiestnenie AP bodov.
14. Dopísanie týchto AP bodov do „.tcl” súboru
15. Vytvorenie testovacieho scenáru v ns-3.
16. Nainštalovanie pohyblivosti a trás z „.tcl” na nodes v ns-3 simulácii
17. Postup s instalaciou veci+++++++++++++++++++++++++++++++++++
18. Spustenie samotnej simulácie
19. Zobrazenie výsledku simulácie v NetAnim.
20. Zobrazenie sieťovej premávky na určených nodes vo Wireshark.
21. Vytvorenie rozhran

### **Implementácia**

Najprv som z Microsoft Store stiahol WSL s Ubuntu 20.04. WSL nakoniec bolo pre túto prácu nevyhnutné, keďže ns-3 poskytuje aj možnosť emulácie pre Linux OS, ale nie Windows. Nasledovalo stiahnutie Display serveru. Potom bolo potrebné stiahnuť SUMO aj so všetkými potrebnými vecami. SUMO som nakoniec vybral kvôli jeho cene (zdarma), možnosti manuálnej tvorby premávky a výberu ľubovoľného miesta na svete pre simuláciu. Ako sieťový simulátor som stiahol ns-3, pretože je zdarma a má výbornú dokumentáciu.

Nasledovalo samotné generovanie scenárov. Najprv sa na ukážku generoval scenár s Račianskym mýtom. Osobne som túto križovatku vybral kvôli jej zložitosti (veľa jazdných pruhov, niektoré s možnosťou odbočenia iné zas nie, veľa semafórov, električky...), veľkosti a blízkosti (ľahké porovnanie vygenerovanej križovatky s reálnym životom). Tu sa zisťovali podrobnosti generovania scenárov v OSM Web Wizard a ich následné použitie a úprava v SUMO. Scenár sa podľa očakávaní podarilo bez problémov vytvoriť. Vygenerovaný scenár obsahoval všetky potrebné časti, všetky cesty, budovy, semafóry a vodorovné značenia boli vytvorené. SUMO potom dokázalo generovať náhodnú premávku. Hneď tu sa, ale vyskytlo niekoľko problémov. Na obrázku je vidieť nesprávne umiestnené semafóry, nesprávne vodorovné značenie a neexistujúce cesty:

A screenshot of a video game

Description automatically generated with medium confidence

Teda každý vygenerovaný scenár bude potrebovať určité úpravy predtým ako bude realistický. Upravovať scenáre do hocijakej podoby je možné buď na začiatku pred vygenerovaním scenáru (tu sa dá nastaviť typ premávky alebo odstrániť zo simulácie všetky inštancie nejakého typu cesty) alebo po vygenerovaní scenáru pomocou množstva “.xml” súborov.

Ďalší sa generoval scenár, ktorý obsahoval okolie fakulty FIIT STU. Tu som chcel otestovať, či je možné simulovať jazdenie auta/áut po cestách v okolí fakulty aby som potom mohol testovať spojenie so sieťovým simulátorom ns-3. Ukázalo sa, že cesty v okolí fakulty nie je možné použiť na premávku áut. Totiž SUMO tieto cesty považoval za cesty pre bicykle a teda nevhodné pre autá, taktiež na nich odmietal generovať premávku áut. Tento problém sa podarilo opraviť úpravou súboru ”osm.net.xml”.

Posledný scenár v SUMO, ktorý sa vytváral bol rovný úsek cesty R7 pri Slovnafte v Bratislave:

A picture containing sky

Description automatically generated

Cieľom tu bolo na rovnom úseku otestovať komunikáciu medzi autami (V2V) a vysielačmi pri diaľnici. Tu sa síce diaľnica na niektorých miestach zdala mať nesprávny počet pruhov, ale vzhľadom na sledovaný cieľ to nebolo podstatné.

Potom bolo načase vytvoriť prvé testovacie scenáre v ns-3. Najprv som prechádzal dokumentáciu. Z nej som sa snažil zreplikovať a spustiť niekoľko tutoriálových scenárov (primárne first.cc a second.cc). Skúšal som sledovať komunikáciu medzi jednotlivými zariadeniami pomocou wireshark a nakoniec som pozoroval ich grafické znázornenie (polohu, komunikáciu) v NetAnim.

Spojenie ns-3 a SUMO je pomerne jednoduché. Len sa do “mobility file” pridá “.tcl” súbor predtým vygenerovaný pre scenár v OSM Web Wizard. Tento súbor obsahuje časové informácie o polohe jednotlivých vozidiel v simulácii. Po spustení simulácie v ns-3 a jej následnej kontrole v GUI sa ukázalo, že sa tu nachádza niekoľko desiatok áut aj napriek tomu, že v SUMO scenári bolo na diaľnici vidieť maximálne 2 autá súčasne. Totiž v “.xml“ súbore s pohybmi áut sa pri každom vozidle nachádza čas kedy sa objaví v SUMO simulácii, keď však ns2mobility vyrába “.tcl” súbor všetky vozidlá sa vytvoria už v čase 0, ale začnú sa pohybovať až v čase určenom “.xml” súborom. Teda vo výsledku je simulácia v NetAnim plná stojacich uzlov, ktoré sa navzájom prekrývajú a začnú sa pohybovať neskôr. Preto som sa nakoniec rozhodol radšej v SUMO vypnúť automatické generovanie premávky (aj tak sa s ním ťažko narába, keďže sa štartovacie pozície/čas pre autá generujú náhodne a ich množstvo je určené počtom na kilometer jazdných pruhov) pre jednoduchosť som si teda manuálne vyklikal 4 autá, ktorým boli ako počiatočné a konečné hrany nastavené okraje diaľnice. Následne som potom sledoval s kódom z tutoriálu “first.cc” komunikáciu medzi pohybujúcimi sa uzlami (autami). Zároveň som sa pokúsil do NetAnim vložiť pozaďový obrázok zo SUMO pre lepšiu ilustráciu simulácie. Toto tak úplne nefungovalo, keďže po vložení sa obrázok zo SUMO nezhodoval s pohybmi áut – autá súradnice v NetAnim sú prehodené oproti SUMO (here) a treba ich externe upraviť.

Následne som na scenári z tutoriálu “third.cc” skúšal rôzne druhy komunikácie medzi uzlami. Point-to-point, csma, wifi... Napokon som to celé pospájal. Autá fungovali ako STA body, ktoré vedeli komunikovať s AP bodmi pri diaľnici. A potom ešte každé auto bolo v podsieti s každým iným autom pričom vždy v každej takejto podsieti bolo jedno auto AP bod a ostatné boli STA. Zároveň som preskúmal možnosť použitia “logarithmic distance propagation loss model” medzi AP a STA v ns-3 teda „Path loss or path attenuation is the reduction in power density of an electromagnetic wave as it propagates through space” [25], ktorá sa počíta v decibeloch a závisí napríklad od vzdialenosti prijímača a vysielača. Hovorí o tom do akej vzdialenosti budú ešte medzi sebou vedieť komunikovať. Použil som “reference distance”, ktorá je bežná v mestskom prostredí ako bolo uvedené v požiadavkach. Tento spôsob predávania si informácii medzi autami síce fungoval, ale potreboval zbytočné množstvo nastavovania a podsietí.

V nasledujucom kroku som chcel zariadiť aby autá medzi sebou nemuseli komunikovať v AP/STA móde, ale ako decentralizované zariadenia teda v Ad-hoc móde “In ad hoc mode, each node transmits data directly to other nodes without access point.” [26] Vo výsledku teda existovali uzly (autá), ktoré sa pohybovali po diaľnici a vždy keď boli v dostatočnej vzdialenosti od seba alebo od vysielačov pri diaľnici dokázali medzi sebou komunikovať. Nakoniec satu nachádzali len 2 podsiete jedna obsahovala vysielače a autá - prijímače (wifi) a druhá len autá (ad-hoc).

Ako ďalší krok bola komunikácia medzi reálnym svetom a simuláciou. Tu som už narazil na limit mojej verzie WSL, používal som WSL 1, ktorá neobsahovala linux kernel. [27] To bol problém keď som chcel použiť linux lxc kontajner na simuláciu reálneho zariadenia na použitie v ns-3 simulácii. [28] Nakoniec som sa však na komunikáciu s ns-3 simuláciou rozhodol použiť tap device vo wsl a teda som nepoužil kontajnery, keďže to bolo jednoduchšie a funkčné riešenie.

Scenár s komunikáciou do simulácie obsahuje to isté ako scenár predtým teda diaľnicu s vysielačmi (wifi), autá (ad-hoc) a teraz sa pridal aj “satelit”, čo je pre simuláciu AP bod (STA body sú vysielače pri diaľnici), ktorý používa range propagation loss model v ns-3, takže propagation závisí len od vzdialenosti prijímača a vysielača pričom sila signálu je rovná sile vysielania až po určitú hranicu a za ňou klesne na nulu. [29] Teda satelit je schopný komunikovať s hociktorým vysielačom pri diaľnici bez straty signálu, pretože som nastavil dosť veľký range. Tento satelit je zároveň takzvaný “ghost node”, ktorý obsahuje TapBridge v simulácii a ten je napojený na Tap device na reálnom zariadení. Ghost node teda slúži výlučne na preposielanie paketov medzi reálnym svetom a ns-3, preto je na ňom vytvorený ešte “párový” uzol, ktorý zabezpečuje napríklad routovanie. [30] Takýmto spôsobom je už možné z linux terminálu pingnúť nejaký uzol v simulácii. Teda ping traverzuje z reálneho rozhrania na linux zariadení cez satelit v simulácii a jeho párový uzol, vysielač pri diaľnici až k požadovanému autu.

Ako posledný krok bolo rozdelenie všetkých vysielačov, áut, satelitov do 2 skupín, ktoré medzi sebou nie sú schopné komunikovať. Napríklad na simulovanie zariadení, ktoré používajú 2G vs 3G... [31] A následná komunikácia zo simulácie pozostávala z vygenerovaných UDP paketov na jednom z áut, ktoré sa posielali na Linux rozhranie.

### **Overenie riešenia**

Scenár 1 – Komunikácia do simulácie s globálnou smerovacou tabuľkou, pohyblivými nodes a jedným AP a STA bodom.

Popis:

Simulácia zameraná na posielanie ping requestu do simulácie z reálneho sveta. Z terminálu bude pingnuté auto - node 13. ICMP pakety teda traverzujú sieť takýmto spôsobom: TAP rozhranie (node 6 a “ghost node “) -> vysielač (node 7 – STA zariadenie) -> vysielač (node 7 – AP zariadenie) -> auto (node 12 – STA zariadenie) -> auto (node 12 – Ad hoc) -> auto (node 13 - Ad hoc). Jedine auto-node-12 je schopné komunikovať s AP bodom (vysielač node 7) pri diaľnici a jedine vysielač node 7 je schopný komunikovať s autami (auto node 12).

Cieľ:

Zistiť, či je možné z terminálu pingnúť node 13, ktorý existuje v simulácii, s použitím globálnej smerovacej tabuľky a pohybujúcich sa nodes.

Spustenie scenáru:

Test:





10.1.6.0 je podsieť kde sa nachádzajú autá, thetap2 je interface cez ktorý chceme komunikovať a gateway je node cez ktorý bude prechádzať komunikácia do simulácie.



Po skončení simulácie je možné pozrieť si “final.txt” a .pcap súbory vo wireshark. Na komunikácii v Netanim je vidieť ako pakety (šípky) smerujú z node 7 do node 12 (Beacon Frames od AP na node 7):

Chart

Description automatically generated

Vo wireshark je však možné vidieť prebiehajúcu komunikáciu na node 5 (10.1.4.1) ghost node, je spojený s tap interface:

Table

Description automatically generated

Na začiatku je vidieť ako najprv hľadá node 6 (10.1.4.2), tent obol určený ako gateway pre vstup do ns-3 siete. Následne potom node 6 hľadá node 7 (10.1.4.3), ktorý už vie komunikovať s autami. A posiela tiež ping request. Ďaľší je obrázok z komunikácie na interface 2 na node 7 (10.1.5.1) a node 12 (10.1.5.2), ktorý je auto:

Table

Description automatically generated

Komunikácia neprebehne, keďže node 7 pošle len 3 ARP requesty a všetky z nich do 15. Sekundy simulácie. Teda žiadne autá sa tam nenachádzajú, komunikácia neprebehne a keďže nedochádza k daľšiemu posielaniu ARP requestov a komunikácia končí.

Záver:

Nie, pri súčasnom nastavení simulácie, pohybujúcich sa nodes a jedno AP/STA nie je možné aby prebehla komunikácia medzi node 13 a reálnym svetom.

Scenár 2 – Komunikácia von zo simulácie s globálnou smerovacou tabuľkou, pohyblivými nodes a jedným AP/STA bodom.

Popis:

Simulácia zameraná na posielanie UDP paketov zo simulácie von na tap rozhranie. V simulácii sa vytvárajú pakety na aute - node 13. Teda pakety traverzujú sieť takýmto spôsobom: auto (node 13 - Ad hoc) -> auto (node 12 – Ad hoc) -> auto (node 12 – STA zariadenie) -> vysielač (node 7 – AP zariadenie) -> vysielač (node 7 – STA zariadenie) -> TAP rozhranie (node 6 a “ghost node “). Jedine auto-node-12 je schopné komunikovať s AP bodom (vysielač node 7) pri diaľnici a jedine vysielač node 7 je schopný komunikovať s autami (auto node 12).

Cieľ:

Zistiť, či je možné posielať pakety vygenerované v simulácii von na rozhranie Operačného systému, ktoré existuje mimo simulácie s použitím globálnej smerovacej tabuľky a pohybujúcich sa nodes.

Spustenie scenáru:

Test:

TAP rozhranie v OS bolo v tomto scenári vytvorené simulátorom ns-3. Po spustení simulácie je možné pomocou wireshark odchytávať premávku na danom rozhraní. Na tomto obrázku sa nachádza časť komunikácie medzi autom (node 13) a TAP rozhraním “vonkajším svetom”:

Table

Description automatically generated

Za povšimnutie stojí, že komunikácia začína zhruba v 45. sekunde. Teda keď sa pozorované autá dostatočne priblížia k AP bodu pri diaľnici. Tento moment je vizuálne znázornený aj pri spustení vygenerovaného súboru “final.xml ” a jeho následnom spustení v Netanim. Kedy si zhruba v 60. sekunde začnú autá a AP bod posielať pakety. V Netanime je tento čas posunutý oproti wireshark, keďže wireshark capture nebol spustený zarovno so simuláciou, ale až po vytvorení TAP rozhrania. Komunikácia v Netanim je skoro identická so scenárom 1, ale šípky smerujú z node 12 do node 7.

Záver:

Áno, je možné zasielať pakety von zo simulácie na rozhranie operačného systému s použitím globálnej smerovacej tabuľky a pohybujúcich sa nodes.

Scenár 3 - Komunikácia do simulácie s globálnou smerovacou tabuľkou, stacionárnymi nodes a jedným AP/STA bodom.

Popis:

Rovnaký ako v scenári 1.

Cieľ:

Zisti, či je možné z terminálu pingnúť node 13, ktorý existuje v simulácii, s použitím globálnej smerovacej tabuľky a stacionárnymi nodes. Pričom sú dané nodes vždy na dosah a preto nehrozí strata ARP requestu.

Spustenie scenáru:



Potom sa rovnako ako v scenári 1 nastaví gateway a urobí sa ping 10.1.6.2.

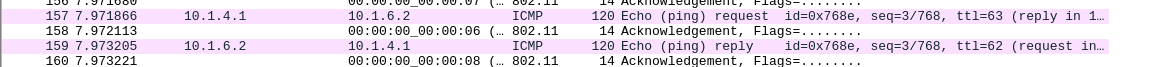
Test:

V Netanim je vidieť rozloženie stacionárnych bodov (autá sa nehýbu). Šípky teraz smerujú aj z node 7 do node 12 a aj naopak.

Chart, scatter chart

Description automatically generated

Vo wireshark je vidieť, že teraz TAP interface (10.1.4.1) dostáva ping reply od auta node 13 (10.1.6.2).



Záver:

Áno, pri súčasnom nastavení simulácie, stacionárnych nodes a jedno AP/STA je možné aby prebehla komunikácia medzi node 13 a reálnym svetom.

Scenár 4 - Komunikácia do simulácie s globálnou smerovacou tabuľkou, pohyblivými nodes a viacerými AP/STA bodmi.

Popis:

Rovnaký ako v scenári 1. Ale teraz je každé auto schopné komunikovať s každým vysielačom zo svojej skupiny.

Cieľ:

Zisti, či je možné z terminálu pingnúť akýkoľvek node, ktorý existuje v simulácii. Zároveň otestovať možnosť škálovania takéhoto riešenia.

Spustenie scenáru:



Test:

Test skončil s errorom a ukazuje sa, že globálne smerovanie nie je úplne najvhodnejšie na wireless siete. Z dokumentácie ns-3: “The model assumes that all nodes on an ns-3 channel are reachable to one another, regardless of whether the nodes can use the channel successfully (in the case of wireless). Therefore, this model should typically be used only on wired topologies.”

Záver:

Na zložitejšie simulácie s wireless sieťou bude vhodnejší OLSR protokol.

Scenár 5 - Komunikácia do simulácie s OLSR smerovacím protokolom, pohyblivými nodes a viacerými AP/STA bodmi.

Popis:

Rovnaký ako v scenári 4. Ale teraz sa namiesto globálnej smerovacej tabuľky použije smerovací protokol OLSR.

Cieľ:

Zisti, či je možné z terminálu pingnúť node 12. S použitím OLSR smerovacieho protokolu.

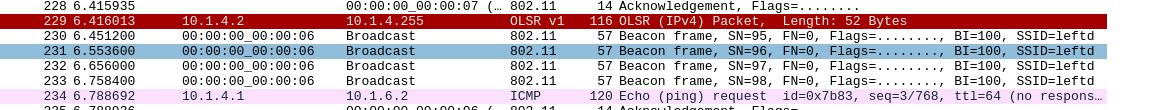
Spustenie scenáru:



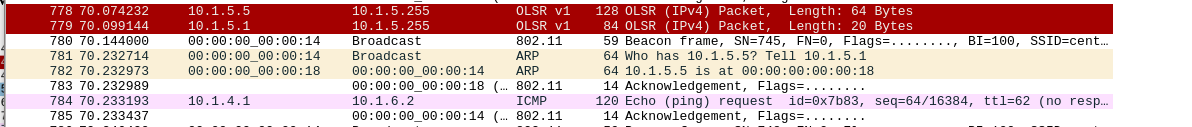
Potom sa rovnako ako v scenári 1 nastaví gateway a urobí sa ping 10.1.6.2.

Test:

Netanim simulácia je veľmi podobná scenáru 1. Rozdiel je len v smere šípok. Vo wireshark je teraz možné vidieť OLSR pakety.



Ďalej je možné sledovať moment kedy auto dorazí na dosah prvému vysielaču v skupine (node 7). Komunikácia na 2. rozhraní node 7 (najprv sa objavia OLSR správy z node 12 (10.1.5.5) a následne sa pošle ARP request a ping request):



Nakoniec je niečo podobné vidieť aj na node 12, ktorý ale negeneruje odpoveď na ARP requesty.

Záver:

Nie, nie je možné takýmto spôsobom a pri takomto nastavení pingnúť node 12, ale je možné cez wireshark sledovať aspoň jednosmernú ICMP komunikáciu a traverzovanie paketov k cielu. Tieto ping requests ale nikdy nedostanú reply.

Scenár 6 - Komunikácia von zo simulácie s OLSR smerovacím protokolom, pohyblivými nodes a viacerými AP/STA bodmi.

Popis:

Rovnaký ako v scenári 2. Ale teraz sa namiesto globálnej smerovacej tabuľky použije smerovací protokol OLSR a viacero AP/STA bodov.

Cieľ:

Zistiť, či je možné posielať pakety vygenerované v simulácii von na rozhranie Operačného systému, ktoré existuje mimo simulácie s použitím OLSR smerovacieho protokolu a pohybujúcich sa nodes.

Spustenie scenáru:



Test:

Vo wireshark na TAP interface je teraz vidieť prichádzajúce OLSR pakety:



Žiadne posielanie UDP paketov však neprebehlo.

Záver:

Áno, na TAP interface boli zachytávané OLSR pakety zo simulácie od rôznych zariadení, ale nepodarilo sa zachytiť UDP pakety z node 12.

## Zhodnotenie

Podarilo sa mi postupovať podľa niekoľkých tutoriálov (here). Vďaka ktorým sa mi podarilo na wsl1 a neskôr podľa postupu (here) na wsl2 rozbehať SUMO a NS-3. Potom som pomocou SUMO dokumentácie poupravoval a vytvoril vlastné scenáre dopravnej simulácie. Neskôr som podľa tutoriálov(here) skombinoval vytvorené simulácie so sieťovým simulátorom NS-3. Sieťovú simuláciu som následne upravoval vďaka NS-3 dokumentácii(here). Pôvodný kód simulácie bol tiež poskladaný z viacerých častí kódov z NS-3 dokumentácie (here). Nakoniec som ešte pomocou NS-3 dokumentácie (here) umožnil komunikáciu medzi vonkajším svetom a simuláciou. Vo výsledku táto práca ukázala, že je možné skombinovať dopravný simulátor SUMO a sieťový simulátor NS-3 a celú simuláciu spustiť na Windows 10 s použitím wsl2, pre jednoduchšie simulácie aj wsl1. Simuláciu je vďaka SUMO možné zasadiť do hocijakého miesta na svete alebo na úplne vymyslené miesta. Testované simulácie umožňujú uskutočniť sledovanie komunikácie medzi viacerými zariadeniami s použitím protokolov ako napr. csma alebo wifi. Komunikáciu je možné sledovať buď s použitím Wireshark alebo vlastného NS-3 GUI - NetAnim. Časti simulácie sú vysoko modifikovateľné a je možné použitie rôznych protokolov, módov wifi zariadení, úprava komunikačných kanálov, úprava vlastností zariadení, vytvorenie vlastných podsietí, UDP/TCP/ICMP komunikácia… V neposlednom rade táto práca ukázala, že je možné posielať pakety z reálneho zariadenia (wsl) do simulácie a tiež zo simulácie na reálne zariadenie.

### **Práca do budúcnosti**

Vyskúšať pomocou nastavení upraviť fungovanie globálneho smerovania tak aby s jeho použitím bolo možné komunikovať aj medzi viacerými AP/STA bodmi s použitím wireless siete.

Otestovať prečo nefunguje ping reply a posielanie UDP paketov mimo simuláciu pri použití OLSR protokolu (je možné, že keďže reálny host negeneruje OLSR pakety, ostatné nodes v simulácii o ňom nevedia, preto sa na TAP rozhraní objavovala len OLSR komunikácia iných nodes, keďže tieto sa posielajú všetkým susedom), možno sa nejako bude dať simulovať posielanie OLSR správ z “ghost node”.

Skúsiť implementovať posielanie viacerých ARP requestov aby sa zabezpečilo, že auto nakoniec tento ARP request nakoniec dostane.

Ns-3 ponúka možnosť simulovať komunikáciu medzi zariadeniami ak je v ceste budova (BuildingsPropagationLossModel). V tejto práci som s budovami nepočítal. Možno by sa dali zariadenia (autá, AP/STA body dali rozdeliť do niekoľkých skupín a takýmto spôsobom simulovať ich komunikáciu ak sú v ceste budovy).

Popis upravy a suborov v SUMO simulacii. Tu alebo este v analyze. Alebo v tech. Dok.

Analyza wireshark

Kod popis v implementacii

Uprava zariadeni v implementacii

Uprava c++ kodu v implementacii

Olsr, ap/sta...

# Literatúra

[1] Krajzewicz, D., Erdmann, J., Behrisch, M. and Bieker, L., 2012. Recent development and applications of SUMO-Simulation of Urban MObility. *International journal on advances in systems and measurements*, *5*(3&4).

[2] Akpaida, V.O.A., Anyasi, F.I., Uzairue, S.I. and Idim, A.I., 2018. Determination of an outdoor path loss model and signal penetration level in some selected modern residential and office apartments in Ogbomosho, Oyo State, Nigeria. *Journal of Engineering Research and Reports*, pp.1-25.

[3] Kotusevski, G. and Hawick, K.A., 2009. A review of traffic simulation software.

**Ad-hoc informácie** –

[4] URL: <https://en.wikipedia.org/wiki/Vehicular_ad_hoc_network>

[5] URL: <https://www.sciencedirect.com/topics/computer-science/vehicular-ad-hoc-network>

**Komunikácia medzi vozidlami -**

[6] URL: <https://en.wikipedia.org/wiki/Vehicular_communication_systems>

**SUMO a jeho aplikácie -**

[7] URL: <https://sumo.dlr.de/docs/SUMO_at_a_Glance.html#included_applications>

**Porovnanie sieťových simulátorov -**

[8] URL: <https://stackoverflow.com/questions/43207692/choosing-between-omnet-or-ns3>

[9] URL: <https://ns3simulation.com/network-simulators-comparison/>

**ns-3 tutoriál a ukážkový kód -**

[10] URL: <https://www.nsnam.org/docs/release/3.7/tutorial/tutorial_18.html>

**Porovnanie simulácie a emulácie -**

[11] URL: <https://www.networkworld.com/article/3227076/network-simulation-or-emulation.html>

**Základné informácie o sieťových simulátoroch -**

[12] URL: <https://en.wikipedia.org/wiki/Network_simulation>

**Virtual machine a WSL –**

[13] URL: <https://www.makeuseof.com/linux-virtual-machine-or-wsl/>

[14] URL: <https://docs.microsoft.com/en-us/windows/wsl/>

[15] URL: <https://en.wikipedia.org/wiki/Windows_Subsystem_for_Linux>

**ns-3 intro -**

[16] URL: <https://www.nsnam.org/about/>

[17] URL: <https://en.wikipedia.org/wiki/Ns_(simulator)>

**inštalácia a požiadavky pre ns-3 -**

[18] URL: <https://www.nsnam.org/wiki/Installation>

**Dopravné simulátory -**

[19]URL: <https://en.wikipedia.org/wiki/Traffic_simulation>

**OSMWebWizard –**

[20] URL: <https://sumo.dlr.de/docs/Tutorials/OSMWebWizard.html>

**TraceExporter -**

[21] URL: <https://sumo.dlr.de/docs/Tools/TraceExporter.html>

**sumo-gui -**

[22] URL: https://sumo.dlr.de/docs/sumo-gui.html

**Netedit –**

[23] URL**:** https://sumo.dlr.de/docs/Netedit/index.html

**NetAnim –**

[24] URL: <https://www.nsnam.org/wiki/NetAnim_3.108>

Display server –

[25] URL: <https://itsfoss.com/display-server/#:~:text=A%20display%20server%20is%20a,your%20computer%20graphically%20(GUI)>.

[26] Li, S., 2019. Comparative Analysis of Infrastructure and Ad-Hoc Wireless Networks. In *ITM Web of Conferences* (Vol. 25, p. 01009). EDP Sciences.

[27] URL: <https://docs.microsoft.com/en-us/windows/wsl/compare-versions>

[28] URL: <https://linuxcontainers.org/>

[29] URL: <https://www.nsnam.org/doxygen/classns3_1_1_range_propagation_loss_model.html#:~:text=Detailed%20Description-,The%20propagation%20loss%20depends%20only%20on%20the%20distance%20(range)%20between,at%20the%20transmit%20power%20level>.

[30] URL: <https://www.nsnam.org/docs/release/3.10/doxygen/group___tap_bridge_model.html>

[31] URL: https://turtler.io/news/Your-2G-Device-Will-Soon-Be-Useless#:~:text=2G%20and%203G%20are%20not,work%20on%20a%203G%20network.

# 

# Príloha – plán práce na projekte



**Týždeň 1 - 3**

Porovnával som rôzne možnosti a simulátory vhodné na analýzu možností výmeny informácií v prostredí navzájom prepojených vozidiel. Najprv som používal len Windows, ale neskôr som sa rozhodol použiť WSL. Skúšal som rôzne postupy a ich kombináciu na inštaláciu všetkého potrebného softvéru. Nakoniec som testoval základné funkcionality SUMO a ns-3.

**Týždeň 3 - 5**

Ako prvú vec som chcel vyskúšať simulovanie okolia univerzity, ale vyskytol sa problém s typom ciest pri škole. Musel som preštudovať ".xml" súbory pre dané scenáre. Potom som skúšal rozsiahlejšiu simuláciu v SUMO s automatickým generovaním vozidiel a následné simulovanie v ns-3. Problém bol, že simulácia bola vďaka automaticky generovaným vozidlám až moc zložitá a neprehľadná, zistil som, že ešte musím preštudovať dokumentáciu k ns-3. Nakoniec som vytvoril len veľmi jednoduchú point-to-point komunikáciu medzi 2 stacionárnymi bodmi.

**Týždeň 5 - 7**

Snažil som sa simulovať komunikáciu medzi pohybujúcimi sa vozidlami a AP bodmi umiestnenými pri R7 v Bratislave. Pokúsil som sa to docieliť pomocou UDP server – klient komunikácie. Servery (AP body) odpovedali na udp pakety poslané klientami (vozidlami). Podarilo sa mi rozpohybovať body v ns-3 simulácii s použitím údajov zo SUMO. Nanešťastie simulácia je v ns-3 prevrátená podĺa y-osi. Čo mi zabránilo pridať obrázok mapy mesta do pozadia. Nakoniec sa mi nepodarilo zabezpečiť aby klient komunikoval s viacerými AP bodmi a prepínal medzi nimi a tiež nebola možná klient – klient komunikácia.

**Týždeň 7 - 9**

Po dlhšom čítaní dokumentácie sa mi podarilo v ns-3 objaviť podporu pre Ad – hoc. Použitie Ad – hoc by mi umožnilo mať komunikáciu medzi vozidlami a AP bodmi pri diaľnici zároveň. Zároveň som počítal minimálne potrebné množstvo AP bodov na vybraný úsek diaľnice tak aby malo vozidlo vždy možnosť komunikovať s AP bodom. Ich vzdialenosť som počítal s použitím logarithmic loss model, vďaka ktorému som získal maximálnu vzdialenosť komunikácie medzi vozidlami v mestskom prostredí. Nakoniec som mal AP body popri R7 a niekoľko vozidiel jazdiacich po nej.

**Týždeň 9 - 11**

Skompletizovanie a spisovanie teoretickej časti bakalárskej práce, príprava na odovzdanie priebežnej správy.