

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. Lukáš Šoltés, PhD.

Máj 2022



ZADANIE BAKALÁRSKEJ PRÁCE

Autor práce: Jakub Ganádik
Študijný program: informatika
Študijný odbor: informatika
Evidenčné číslo: FIIT-5212-102916
ID študenta: 102916
Vedúci práce: Ing. Lukáš Šoltés, PhD.
Vedúci pracoviska: Ing. Katarína Jelemenská, PhD.

Názov práce: **Analýza možností výmeny informácií v prostredí navzájom prepojených vozidiel.**

Jazyk, v ktorom sa práca
vypracuje: slovenský jazyk

Špecifikácia zadania: 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 situácii 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 navrhnete simulačný nástroj alebo navrhnete a implementujete metódu overenia riešenia v reálnom prostredí. Overenie vykonajte prostredníctvom simulácie, alebo reálneho merania. Výsledky zhodnoťte a porovnajte s inými publikovanými prácami.

Rozsah práce: 40

Termín odovzdania
práce: 16. 05. 2022

Dátum schválenia
zadania práce: **23. 11. 2021**

Zadanie práce schválil: **doc. Ing. Valentino Vranič, PhD.**
garant študijného programu

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

V Bratislave, 16.5.2022

Jakub Ganádik

Pod'akovanie

Chcel by som pod'akovať svojmu školiteľovi Ing. Marekovi Galinskému, PhD. Za pomoc, rady a usmernenie pri písaní mojej práce.

Ďalej by som chcel pod'akovať svojej rodine za psychickú a finančnú podporu počas písania tejto práce.

Anotácia

Slovenská technická univerzita v Bratislave

FAKULTA INFORMATIKY A INFORMAČNÝCH TECHNOLOGIÍ

Študijný program:	Informatika
Autor:	Jakub Ganádik
Bakalárska práca:	Analýza možností výmeny informácií v prostredí navzájom prepojených vozidiel
Vedúci bakalárskej práce:	Ing. Lukáš Šoltés, PhD.

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.

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.

Obsah

1	Úvod.....	1
2	Analýza.....	3
2.1	Simulátor dopravy	3
2.2	Sieťový simulátor	3
2.3	SUMO	4
2.4	Použíte nástroje v SUMO.....	4
2.5	SUMO v porovnaní s inými simulátormi dopravy	8
2.6	Tap device	9
2.7	Waf	9
2.8	ns-3	9
2.9	Použíte triedy.....	11
2.10	Príklady využitia ns-3 a SUMO zo skutočného života.....	15
2.11	Wireshark	16
2.12	Windows subsystem for Linux.....	17
3	Opis riešenia.....	21
3.1	Špecifikácia požiadaviek.....	21
3.2	Návrh riešenia	24
3.3	Implementácia	26
3.4	Overenie riešenia.....	37
4	Zhodnotenie.....	53
4.1	Práca do budúcnosti	53
	Literatúra	21
	Príloha A: plán práce na projekte	A-1
	Príloha B: postup inštalácie.....	B-1
	Príloha C: Popis kódu.....	C-1

Zoznam obrázkov

Obrázok 1 OSM Web Wizard Staré Mesto	5
Obrázok 2 Sumo-gui Staré Mesto.....	7
Obrázok 3 Netedit tvorenie simulácie	8
Obrázok 4 NetAnim simulácia ukážka	10
Obrázok 5 LogPropagationLossModel vzorec	12
Obrázok 6 Prepojenie Linux OS a ns-3	15
Obrázok 7 Web Wizard Račianske mýto.....	27
Obrázok 8 označené chyby vo vygenerovanom scenári.....	27
Obrázok 9 chýbajúca križovatka	28
Obrázok 10 chýbajúci semafor	28
Obrázok 11 možnosť úpravy SUMO simulácie	29
Obrázok 12 úprava existujúceho scenáru v NetEdit.....	30
Obrázok 13 okolie FIIT STU v SUMO	30
Obrázok 14 ukážka uzlov pri automatickom generovaní	31
Obrázok 15 ukážka tcl súboru	32
Obrázok 16 SUMO R7 v Bratislave	32
Obrázok 17 trajektória pohybu vozidiel 1	33
Obrázok 18 trajektórie pohybu vozidiel 2	33
Obrázok 19 NetAnim finálna podoba vozidlá/vysielače	35
Obrázok 20 NetAnim finálna verzia podsiete	36
Obrázok 21 NetAnim úprava simulácie	37
Obrázok 22 NetAnim sledovanie animácie	38
Obrázok 23 TAP rozhrania	39
Obrázok 24 UDP generátor	40
Obrázok 25 UDP generátor Wireshark.....	40
Obrázok 26 vygenerované pcap súbory	41
Obrázok 27 ping 1	42
Obrázok 28 Wireshark komunikácia 1	42
Obrázok 29 ping 2	43
Obrázok 30 Wireshark komunikácia 2	43
Obrázok 31 Wireshark komunikácia 3	43
Obrázok 32 Wireshark komunikácia 4	43
Obrázok 33 Wireshark OLSR capture	44
Obrázok 34 Wireshark OLSR správy	44
Obrázok 35 Wireshark OLSR správy 2	45
Obrázok 36 NetAnim pohyblivé uzly animácia	46
Obrázok 37 Wireshark komunikácia 5	46
Obrázok 38 Wireshark komunikácia 6	47
Obrázok 39 NetAnim znázornenie routovacích ciest	49
Obrázok 40 Wireshark komunikácia 7	50
Obrázok 41 Wireshark ping 3.....	51
Obrázok 42 Wireshark komunikácia loopback.....	51
Obrázok 43 Wireshark ping 4.....	51
Obrázok 44 Wireshark komunikácia 8	52

1 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 medzi vozidlovú 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 dáta, hlavne bezpečnostné upozornenia a dopravné informácie. Tieto informácie môžu slúžiť na vyhýbanie sa dopravným nehodám alebo zmiernenie kolón. Komunikácia medzi vozidlami je zvyčajne vyvíjaná ako súčasť inteligentných dopravných systémov. Začiatky ich vývoja sa datujú späť do 70. rokov minulého storočia. Na prenos dát boli používané rôzne 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 uzly (nodes) vo VANET-e a sieťový simulátor, ktorý by simuloval bezdrôtovú sieť. [1] [2] [3] [4]

V tejto práci sa budem sústreďiť 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 testovať možnosť komunikácie reálnych zariadení so simuláciou.

2 Analýza

2.1 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 akademických inštitúcií ich používa pri simulovaní dopravných situácií. 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 kategórií:

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]

2.2 Sieťový simulátor

2.2.1 Sieťová simulácia

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 diskretných 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. [6] [7]

2.2.2 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 posielat' skutočné pakety zo živej aplikácie cez simuláciu virtuálnej siete. Skutočný paket sa "moduluje" do simulačného paketu. [7]

2.3 SUMO

"Simulation of Urban Mobility" (SUMO) je mikroskopický a open source simulátor. Umožňuje 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 semaforov 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]

2.4 Použité nástroje v SUMO

2.4.1 SUMO

Samotná simulácia, ktorá je kontinuálna v priestore a diskretná 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 semaforey, cestné značenie... Tieto sú celkom vhodné keďže SUMO má niekedy problém určiť správnu pozíciu semaforov na mapách vygenerovaných OSM Web

Wizard. Vizualizácia cestnej siete sa uskutočňuje pomocou sumo-gui. Pomocou SUMO môžeme získať výstup vo veľa rôznych formátoch. Ja budem používať “.xml” formát. [8]

2.4.2 OSM Web Wizard

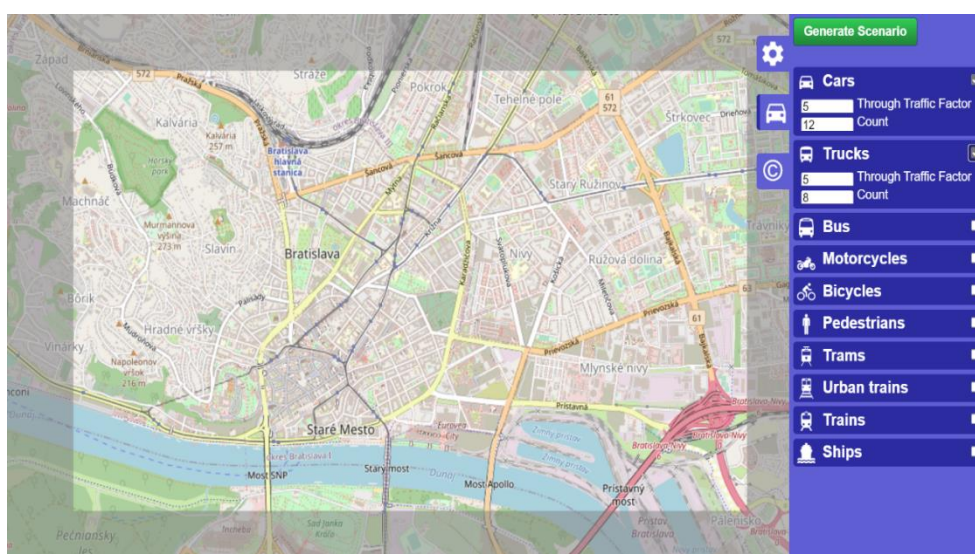
OSM Web Wizard ponúka jedno z najjednoduchších a najviac priamočiarych riešení, ako začať so SUMO. Vďaka OSM Web Wizard je možné vybrať časť mapy z openstreetmap a následne na nej nakonfigurovať náhodné požiadavky na premávku a spustiť a vizualizovať scenár v sumo-gui.

Po spustení je možné vybrať si akékoľvek miesto na svete. Zaškrtnutím Select Area sa začína výber požadovanej oblasti. Kliknutím na tlačidlo “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 na rozdiel od hrán na okraji.

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



Obrázok 1 OSM Web Wizard Staré Mesto

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 ďalšiu 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 (zide sa obzvlášť v prípadoch kedy sú vo vygenerovanej simulácii niektoré cesty chybne označené). [9]

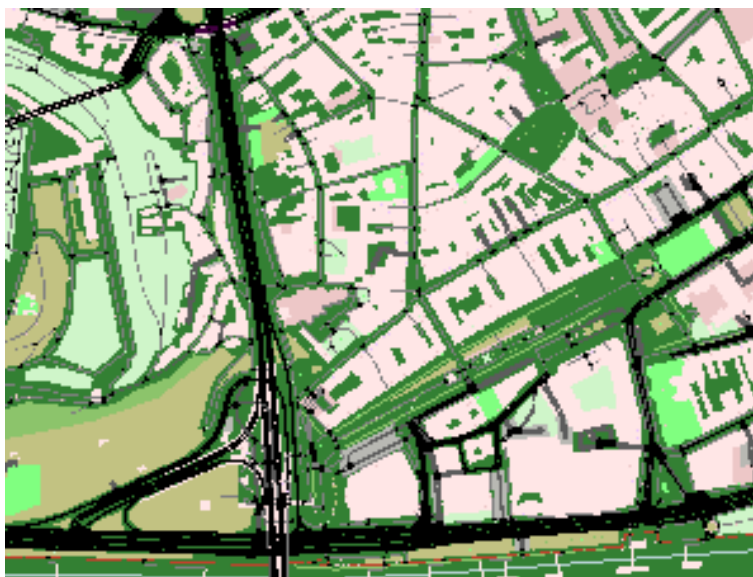
2.4.3 *TraceExporter*

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 ďalšie dôležité informácie pre každé vozidlo v každom momente. Výstup by sa dal interpretovať ako super presné vysokofrekvenčné GPS zariadenie pre každé vozidlo. [10]

2.4.4 *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 semaforov, sledovať vozidlá, meniť nastavenia cestného značenia, jazdných pruhov, križovatiek.

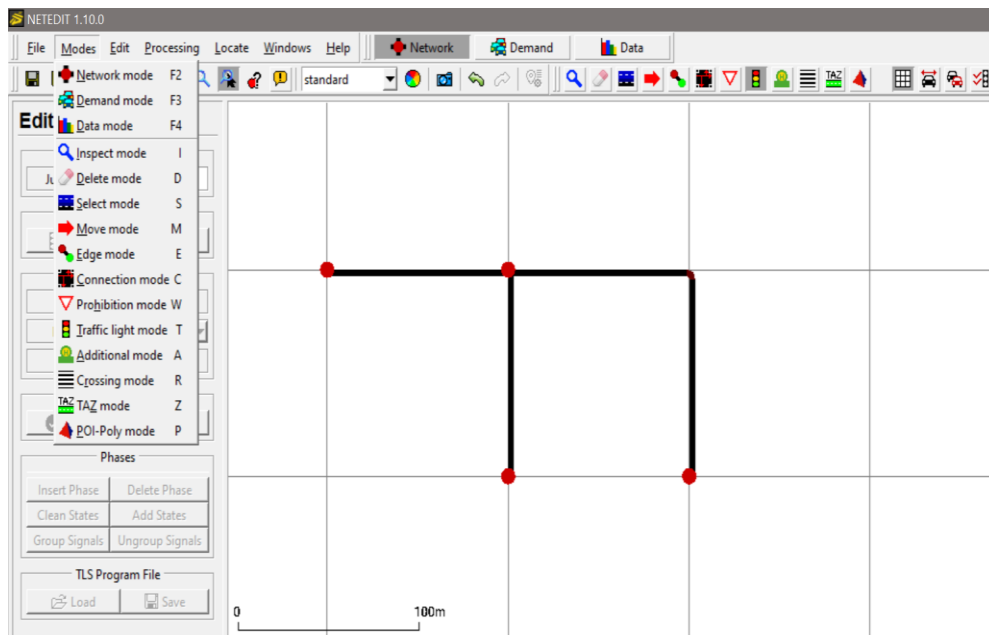
Nachádza sa tu aj menu na zmenu farieb vďaka nemu môžeme napríklad meniť veľkosti a farbu vozidiel. [11] Ukážka scenáru v sumo-gui (Staré mesto):



Obrázok 2 Sumo-gui Staré Mesto

2.4.5 Netedit

Je vizuálny sieťový editor. V Netedit vieme pridávať nové cesty, križovatky, semaforey... 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.



Obrázok 3 Netedit tvorenie simulácie

Ukážka Netedit, tvorenia hrán a rôznych módov, ktoré sa tu nachádzajú. [12]

2.5 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 na rozdiel 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. [13]

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

2.6 Tap device

TAP Poskytuje príjem a prenos paketov pre programy v user space. Je to jednoduché zariadenie typu Point-to-Point alebo Ethernet, ktoré namiesto prijímania paketov z fyzického média ich prijíma z user space programu a namiesto odosielania paketov cez fyzické médium ich zapisuje do user space programu. [14]

2.7 Waf

Slúži na kompiláciu zdroja do podoby použiteľného programu. Waf je build system, ktorý sa používa na ns3 projekty. Je napísaný v pythone. [15]

2.8 ns-3

2.8.1 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. [16] [17]

2.8.2 *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 uzly, 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 uzlami potrebujeme urobiť linku a na to použijeme tzv. Helper (PointToPoint, Csma...), ktorá prepojí pomocou komunikačného kanálu 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 uzol STA/AP/Ad hoc. Potom sa používa tiež InternetStackHelper, ktorý nainštaluje Internet stack teda IP, UDP, TCP ... na každý uzol v predtým určenom kontajnery. Jednotlivým zariadeniam potom môžeme pomocou Ipv4AddressHelper priradiť ip adresu z určeného rozsahu. Nakoniec môžeme z vybraných uzlov vytvoriť generátor paketov a posielat' pakety na určený port určeného uzlu. [18] [19]. V prípade, že by sme chceli posielat' pakety z reálneho zariadenia do simulácie alebo zo simulácie na skutočné rozhranie, bude treba vytvoriť TapBridgeHelper a spojiť ho s reálnym TAP rozhraním. Na vizualizáciu daného scenáru používam NetAnim a Wireshark.

2.8.3 NetAnim

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



Obrázok 4 NetAnim simulácia ukážka

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. [20]

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 bolo potrebné nainštalovať ns-3 na Windows Subsystem for Linux. [19]

2.8.4 *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 na rozdiel od ns-3 prichádza s už vstavaným interface-om 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 a dokumentácia pri ktorej sa zdá, že je lepšia pri ns-3. Nakoniec som sa práve kvôli veľkému množstvu tutoriálov, dokumentácie a videí rozhodol použiť ns-3. [21] [22]

2.9 Použité triedy

2.9.1 *ns3::YansWifiChannelHelper Class*

“Manage and create wifi channel objects for the YANS model.” [23]

S pomocou tejto triedy vieme vytvoriť komunikačné kanále. Ich počet závisí od frekvencie, ktorú používa router. Je to časť z dostupných frekvencií na ktorých sa môže komunikovať. [24]

Kanálu dokážeme nastaviť viacero atribútov. Napríklad propagation delay, ktorý je definovaný ako “flight time of packets over the transmission link and is limited by the speed of light. For

example, if the source and destination are in the same building at the distance of 200 m, the propagation delay will be $\sim 1 \mu\text{sec}$.” [25]. Defaultne je v ns-3 nastavená `constantspeedpropagationdelay`, teda rýchlosť propagácie je konštanta a default hodnota je rýchlosť svetla vo vákuu v m/s. Ďalej sa tu dá upravovať `propagationloss`, ktorá modeluje postupnú stratu sily elektromagnetickej vlny keď sa šíri v priestore a cez rôzne predmety. Teda v závislosti od predmetu, ktoré stoja v ceste postupne slabne el. vlna vychádzajúca od vysieláča a preto je možné, že nebude schopný komunikovať s prijímačom. Ns-3 ponúka viacero typov. Napríklad `RangePropagationLossModel` závisí len od vzdialenosti vysieláča a prijímača. Maximálna vzdialenosť (`range`) sa udáva v metroch, všetky prijímače do tejto vzdialenosti budú vysielanie dostávať so silou aká je pri vysieláči. Všetky mimo tejto vzdialenosti ho budú dostávať so silou -1000 dbm, teda 0. Ďalší veľmi dôležitý model je `LogDistancePropagationLossModel`, ktorý počíta silu prijímania a dá sa vypočítať podľa vzorca [26] :

$$PL(dBm) = PL(d_0) + 10n \log \frac{d}{d_0}$$

Obrázok 5 `LogPropagationLossModel` vzorec

Kde n je path loss distance exponent s defaultnou hodnotou 3

d je vzdialenosť vysieláča a prijímača

d_0 je reference distance default je 1m

$PL(d_0)$ je path loss (propagation loss) vo vzdialenosti d_0

Ak je path loss menšia ako reference distance, vráti sa sila rovnaká ako pri vysieláči. [27]

Posledný propagation loss model, ktorý stojí za spomenutie je `BuildingsPropagationLossModel`, ktorý simuluje straty v prípade tienenia (vnútri aj vonku), prechodu vonkajšej steny aj vnútornej steny. Teda by sa takto mohla dať predstierať prítomnosť budov. [28]

2.9.2 *ns3::YansWifiPhyHelper Class*

“Make it easy to create and manage PHY objects for the YANS model.” [29]

Pomocou tejto triedy dokážeme nastaviť nášmu kanálu typ fyzickej vrstvy akú by mali zariadenia v tomto kanáli používať.

2.9.3 *ns3::WifiMacHelper Class*

“create MAC layers for a ns3::WifiNetDevice.”

Pričom MAC je definovaná ako: “The part of the data link protocol that controls access to the physical transmission medium in local networks”. [30]

WifiMacHelper dokáže vytvoriť niekoľko MAC typov: AP, STA a Ad-Hoc. [31]

AP (Access point) zariadenie: Access point je zariadenie, ktoré vytvára vo svojom okolí bezdrôtovú lokálnu sieť (WLAN). AP sa väčšinou pripojí k router-u, switch-u alebo hub-u cez ethernetový kábel a vysiela wifi signál danej oblasti. Namiesto kupovania router-u na miesta kam už signál z neho nedosiahne, stačí natiahnuť ethernetový kábel a pripojiť access point. [32]

STA (wifi station): je hocijaké zariadenie, ktoré sa vie pripojiť k sieti. Access point-y vedľa fungovať ako hub pre jedno alebo viac zariadení STA aby im poskytli pripojenie na internet. [33]

Ad-Hoc: ad-hoc sieť sa vytvára spontánne keď sa zariadenia pripoja a komunikujú spolu. Ad-hoc siete bývajú väčšinou LAN siete. Zariadenia medzi sebou komunikujú bez toho aby potrebovali nejakú base station alebo access point na koordináciu komunikácie. Každé zariadenie sa podieľa na routovaní, zistí správnu routu použitím routovacieho algoritmu a následne posiela dáta ostatným zariadeniam. [34]

2.9.4 *ns3::Ipv4GlobalRoutingHelper Class*

Vytvára Ipv4GlobalRouting objekty. Ipv4GlobalRouting je globálny routovací protokol pre IPv4 stack. Tu sa dá úpravou atribútov nastaviť prepočítavanie globálnych routy. Ak sú medzi dvoma uzlami dve cesty s rovnakou cenou, dá sa nastaviť preposielanie paketov oboma linkami súčasne alebo len použitie jednej z nich stále. [35]

2.9.5 *ns3::InternetStackHelper Class*

Táto trieda umožňuje pcap a ascii sledovanie udalostí v internet stack-u asociovaným s určitým uzlom. Tu sa dá nastaviť ak by sme chceli používať custom routing protocol napr. OLSR na určitom uzle. [36]

2.9.6 *ns3::OnOffApplication Class*

Vytvára premávku do jednej destinácie s použitím onoff vzoru. “On” a “Off” stavy sa striedajú, dĺžka každého z týchto stavov je určená časom vypnutia a zapnutia. Počas stavu vypnutia sa negeneruje žiadna premávka. Pri stave zapnutia sa premávka generuje a je špecifikovaná “data rate” defaultne 500000bps a “packet size” defaultne 512. Následne sa tu dá nastaviť IP adresa destinácie paketov a port na ktorý sa budú tieto pakety posielat’. [37]

2.9.7 *ns3::TapBridge Class*

Bridge, vďaka ktorému to vyzerá, že reálne zariadenie je pripojené na ns-3 zariadenie.

Ns-3 a spojenie s reálnym svetom pomocou Tap Bridge

Tap Bridge žije v akomsi sivom svete niekde medzi reálnym zariadením s Linuxom a zariadením bridge-u ns-3. Z pohľadu Linuxu sa tento kód javí ako obslužný program user módu pre zariadenie Tap. To znamená, že keď na Linuxe zapisujeme do zariadenia /dev/tap (ktoré je buď manuálne alebo automaticky vytvorené v závislosti operačného režimu), zápis je presmerovaný do TapBridge, ktorý žije vo svete ns-3. Inými slovami, linux proces zapíše paket do TAP zariadenia a tento paket je presmerovaný na proces ns-3, kde ho prijme TapBridge ako výsledok operácie čítania. TapBridge potom pošle paket do sieťového zariadenia ns-3, ku ktorému je premostený. V opačnom smere je paket prijatý sieťovým zariadením ns-3 premostený do TapBridge. TapBridge potom vezme tento paket a zapíše ho späť na Linux TAP. Tento zápis do zariadenia sa potom OS javí, ako keby na jeho zariadenie prišiel paket. [38]

Ns-3 obsahuje tri operačné režimy pre TAP zariadenia: ConfigureLocal UseLocal a UseBridge.

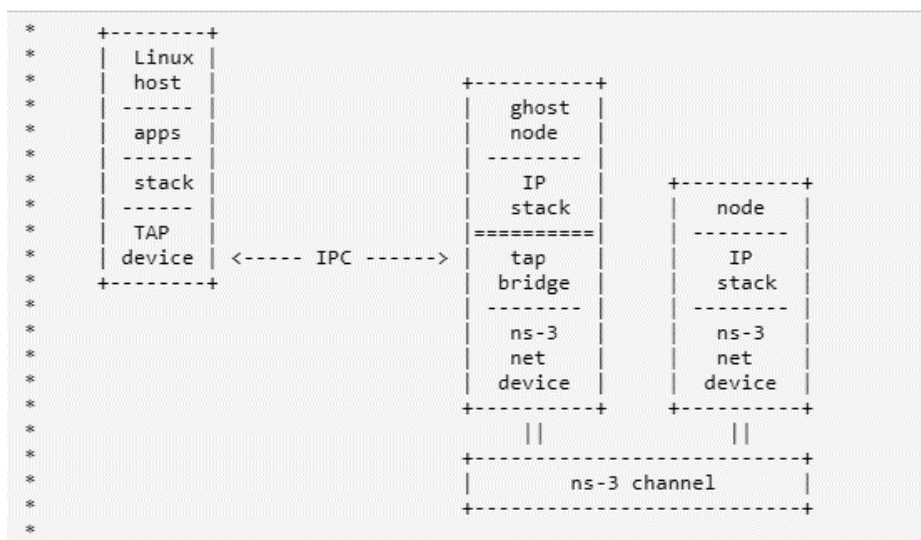
ConfigureLocal mód:

V režime ConfigureLocal je konfigurácia TAP zariadenia zameraná na konfiguráciu ns-3. Informácie o konfigurácii sa prevezmú zo zariadenia v simulácii ns-3 a automaticky sa vytvorí TAP zariadenie zodpovedajúce atribútom ns-3. V tomto prípade počítač s Linuxom vyzerá, akoby bol priamo pripojený k simulovanej sieti ns-3.

V tomto prípade sa “sieťové zariadenie ns-3” v “ghost node” javí tak, ako keby skutočne nahrádzalo zariadenie TAP v Linuxe. Simulácia ns-3 vytvorí zariadenie TAP na Linux OS a nakonfiguruje adresy IP a MAC zariadenia TAP tak, aby sa zhodovali s hodnotami priradenými

k simulovanému zariadeniu siete ns-3. Celé usporiadanie pôsobí ako most, ale medzi zariadeniami, ktoré majú identické IP a MAC adresy.

Tu sa od používateľa nevyžaduje, aby poskytoval žiadne konfiguračné informácie špecifické pre TAP. TAP vytvorí a nakonfiguruje ns-3 podľa jeho predvolených nastavení, vrátane mena, ktoré sa dá nastaviť v ns-3 a podľa neho toto zariadenie vieme nájsť na Linux OS. Režim ConfigureLocal je predvoleným prevádzkovým režimom Tap Bridge. Obrázok prepojenia Linux OS a ns-3 simulácie s použitím ConfigureLocal mode:



Obrázok 6 Prepojenie Linux OS a ns-3

UseLocal mód:

Je podobný predchádzajúcemu módu, ale teraz majú TAP device a jeho ns-3 zariadenie inú MAC adresu a TAP zariadenie je potrebné vytvoriť samostatne v Linux OS, nie cez ns-3.

UseBridge mód:

Ako predchádzajúci mód, ale tu si používateľ vytvorí aj vlastný bridge. [39]

2.10 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 infraštruktúrou. 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 čelí 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 podsystemy. 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 ďalšie 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]

2.11 Wireshark

Wireshark je celosvetovo najpoužívanejší analyzátor sieťových protokolov. Umožňuje vidieť, čo sa deje v sieti na mikroskopickej úrovni a je de facto štandardom v mnohých komerčných a neziskových podnikoch, vládnych agentúrach a vzdelávacích inštitúciách.

Wireshark má kopu vlastností napr.:

Možnosť hĺbkovej inšpekcie stoviek protokolov a neustálym pridávaním nových

Live capture a offline analýza

Zachytené sieťové údaje je možné prehliadať pomocou GUI

Najvýkonnejšie zobrazovacie filtre v tomto odvetví

Čítanie/zápis mnohých rôznych formátov zachytávaných súborov napr.: tcpdump, Pcap NG...

Dáta je možné čítať z Ethernetu, IEEE 802.11, PPP/HDLC, ATM, Bluetooth, USB, a ďalších

Výstup je možné exportovať do formátu XML a iných. [40]

2.12 Windows subsystem for Linux

2.12.1 WSL

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. [41] [42] Bol vytvorený spoločnosťou Microsoft a z Microsoft store sa dá nainštalovať Ubuntu 20.04. V súčasnosti existujú 2 verzie WSL1 a WSL2. Hlavné dôvody prečo si vybrať wsl2 namiesto wsl1: Zvýšený výkon file systému a podporuje plnú kompatibilitu systémových volaní.

WSL2 tiež používa virtualizačnú technológiu na rozbehávanie Linux kernelu vnútri jednoduchšieho lightweight virtual machine.

WSL2 však nie je typické VM, pretože nepotrebuje bežať všetky procesy, ktoré bežia na skutočnom VM. [43] Tabuľka najdôležitejších rozdielov:

Feature	WSL 1	WSL 2
Integration between Windows and Linux	✓	✓
Fast boot times	✓	✓
Small resource foot print compared to traditional Virtual Machines	✓	✓
Runs with current versions of VMware and VirtualBox	✓	✓
Managed VM	×	✓
Full Linux Kernel	×	✓
Full system call compatibility	×	✓
Performance across OS file systems	✓	×

Obrázok 7 WSL1 vs WSL2

Vzhľadom na to, že WSL2 má oveľa viac schopností ako WSL1, rozhodol som sa použiť WSL2.

2.12.2 *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 “. [19]

2.12.3 *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 ďalšieho 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. [44]

2.12.4 *Xming*

Je to display server pre operačné systémy Windows. Server spracuje požiadavky a zobrazí ich pre používateľa ako grafický výstup. [45]

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

3 Opis riešenia

3.1 Špecifikácia požiadaviek

3.1.1 *Cieľ*

Nájsť a otestovať simulačné prostredie/prostredia vďaka ktorým by som mohol simulovať komunikáciu medzi vozidlami a okolím. Simulačný 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.

3.1.2 *Funkcionálne požiadavky*

1. Všetky simulácie by mali byť spustiteľné na mojom zariadení.
2. WSL by malo byť schopné 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, semaforey...
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/emulácie.

10. Sieťový simulátor by mal mať možnosť upraviť simulačný 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 wifi by mali mať všetky uzly možnosť byť STA/AP/Ad-Hoc.
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 posielat' pakety zariadeniam v simulácii.
27. Zariadenia v simulácii by mali byť schopné posielat' 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.

3.1.3 *Nefunkcionálne požiadavky*

1. Sieťová premávku by malo byť možné zachytiť pomocou network protocol analyzeru najlepšie by bolo použiť wireshark.
2. Semafore, 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 posilať a rozumiť 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 d_0 of 10 m or 100 m is recommended." [26]
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 posilať beacon frames každému zariadeniu STA.
10. Jednotlivé uzly by mali byť schopné fungovať ako access point, station a ad-hoc. Ideálne naraz v rôznych podsieťach.
11. Uzol generujúci pakety by mal byť schopný poslať pakety na určenú IP adresu a port.
12. Simulátor by mal podporovať globálne routovanie pre menšie simulácie.
13. Simulátor by mal podporovať OLSR routovací protokol.
14. Uzly používajúce OLSR by mali generovať "hello", "topology control" a "MID" správy.
15. Všetky simulácie by sa mali dať spustiť na x64 Windows 10.
16. Simulácie by mali bežať na windows subsystem for linux 1 alebo 2.
17. WSL by malo používať display server na spustenie GUI.
18. Vozidlá by mali byť schopné medzi sebou komunikovať v režime ad hoc.
19. Wireshark by mal byť schopný zachytiť pakety smerujúce zo simulácie na rozhraní môjho zariadenia.
20. Mal by som byť schopný pingnúť zariadenia v simulácii zo svojho zariadenia.

3.2 Návrh riešenia

1. Úplne ako prvú vec budem musieť vyriešiť problém spustiteľnosti na mojom zariadení.
Vzhľadom na to, že väčšina ns-3 tutoriálov je na Linux a ns-3 dokumentácia hovorí, že je lepšie použiť WSL ak už nie je iná možnosť a je treba pracovať na Windows.
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 otestovanie schopností SUMO, do úvahy pripadá na testovacie účely Račianske mýto, keďže je to zložitá a frekventovaná križovatka.
6. Vytvorenie scenáru z vybraného miesta z open maps. Na to sa použije OSM Web Wizard.
7. Následne bude treba upraviť daný vygenerovaný scenár. Môže sa napríklad stať, že semaforey 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 uzloch v ns-3 simulácii
17. Vytvorenie sieťových zariadení z uzlov
18. Vytvorenie AP/STA/Ad-Hoc wifi zariadení

19. Vytvorenie komunikačných kanálov
20. Úprava vlastností komunikačných kanálov a samotných zariadení
21. Príprava simulácie na použitie viacerých routovacích protokolov
22. Vytvorenie aplikácie na posielanie paketov na určitú IP a port
23. Použitie tried na prepojenie simulácie s reálnym svetom
24. Spustenie samotnej simulácie
25. Zobrazenie výsledku simulácie v NetAnim.
26. Zobrazenie sieťovej premávky na určených uzloch vo Wireshark.

3.3 Implementácia

Najprv som z Microsoft Store stiahol WSL s Ubuntu 20.04, tento krok som spravil ešte pred začatím tejto práce a v tejto časti som pracoval s WSL1. WSL nakoniec bolo pre túto prácu nevyhnutné, keďže ns-3 poskytuje aj možnosť emulácie pre Linux OS, ale nie Windows. V tejto časti sa inštalovali rôzne package potrebné na ďalšiu prácu s WSL napr.: build-essentials, net-tools, taskel a ďalšie.

Nasledovalo stiahnutie Display serveru. Po stiahnutí display server už bolo možné ostatné veci implementovať priamo vo WSL. Ďalej treba nainštalovať ďalšie package potrebné pre SUMO. SUMO som neinštaloval pomocou “install SUMO”, ale kvôli lepšej modifikovateľnosti si ho naklonujem z github. 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 a build som urobil pomocou “build.py”. Ns-3 som vybral, pretože je zdarma a má výbornú dokumentáciu z ktorej sa dalo vyčítať, že ns-3 by malo spĺňať väčšinu požiadaviek na sieťový softvér.

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 semaforov, 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. V OSM Web Wizard som generoval scenár na základe tejto mapy:



Obrázok 8 Web Wizard Račianske myto

Vygenerovaný scenár obsahoval všetky potrebné časti, všetky cesty, budovy, semaforey a vodorovné značenia boli vytvorené. SUMO potom dokázalo generovať náhodnú premávku. Hneď tu sa, ale vyskytlo niekoľko problémov. Po vygenerovaní som mal podozrenie na nesprávne umiestnené semaforey a neexistujúce cesty:



Obrázok 9 označené chyby vo vygenerovanom scenári

Tu som využil blízkosť tejto križovatky a išiel som svoje podozrenie skontrolovať.

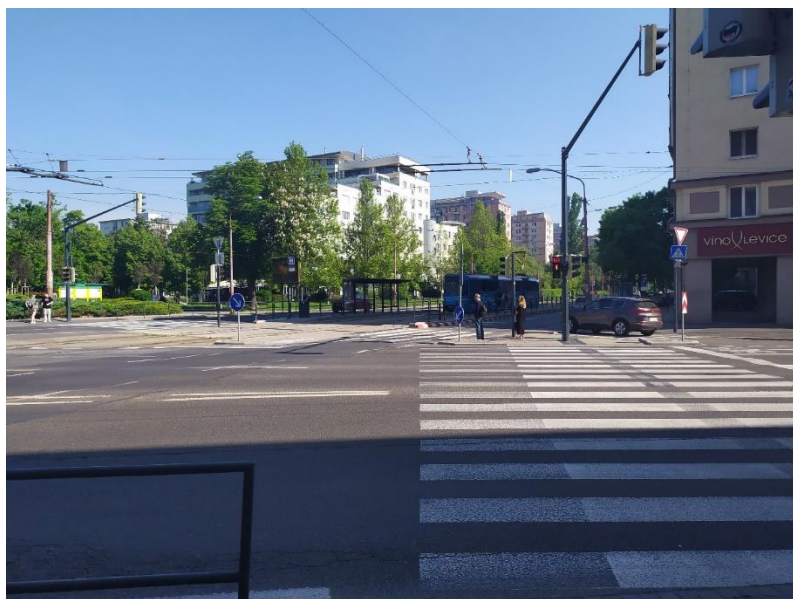
Na čísle 1 sa nachádza neexistujúca cesta a križovatka:



Obrázok 10 chýbajúca križovatka

Cesta a križovatka sa tu síce v nejakej podobe nachádzajú, ale určite nie tak ako to vygeneroval SUMO. Pravdepodobne to vychádza zo spôsobu akým je táto “cesta” označená v OSM Web Wizard, kde vyzerá ako menšia cesta, tak sa ju SUMO snažilo spojiť so skutočnou cestou vedľa, čo viedlo k vytvoreniu tejto križovatky.

Na číslach 2 sa nachádzajú neexistujúce semaforey:

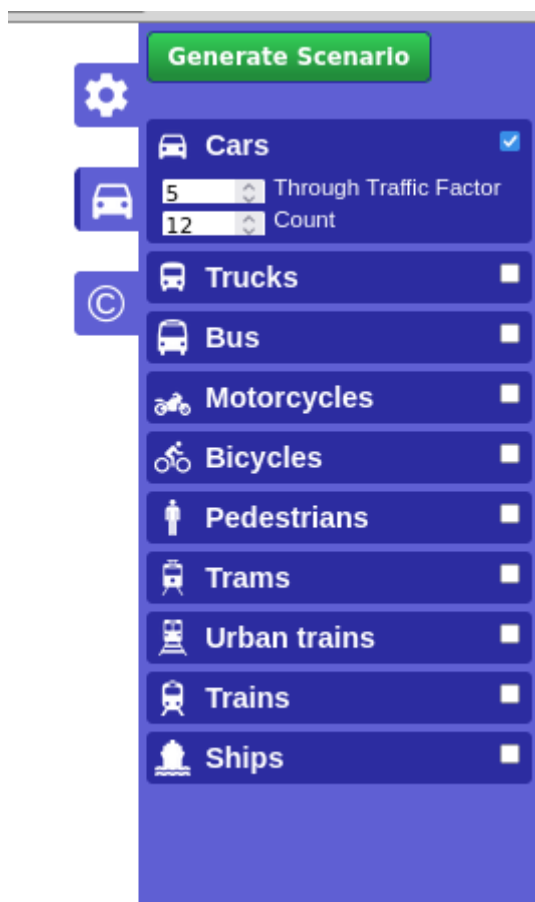


Obrázok 11 chýbajúci semafor

SUMO umiestnilo niektoré semaforey na opačné strany križovatiek. Nie je to, ale kvôli tomu, že by SUMO používalo americké semaforey, pretože ostatné, ktoré som skontroloval sú na správnych miestach. Umiestnenia semaforov sa z nejakého dôvodu nezhodujú ani s piktogramami semaforov z OSM Web Wizard.

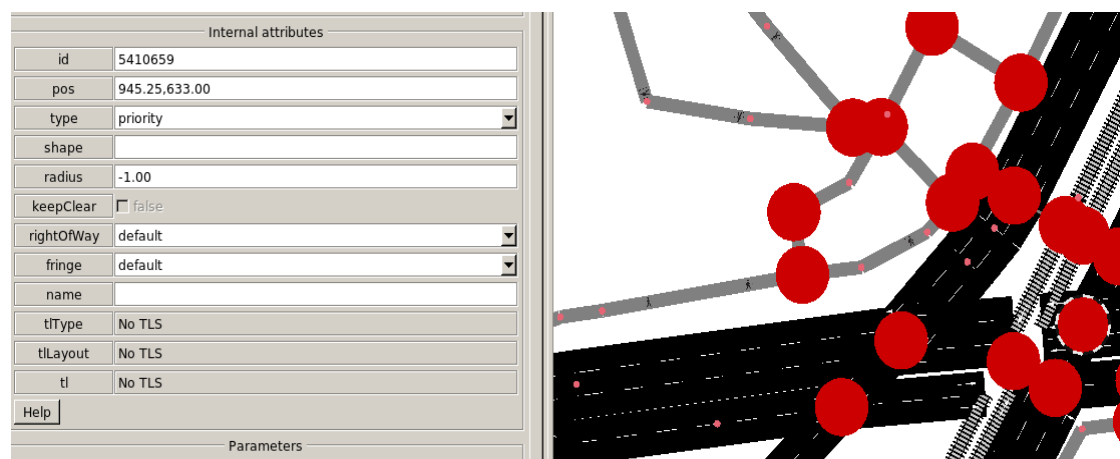
Vo výsledku teda bude každý vygenerovaný scenár potrebovať určité úpravy predtým ako bude realistický. Upravovať scenáre do hocijakej podoby je možné buď na začiatku pred vygenerovaním scenáru (tu sa dá nastaviť typ premávky alebo odstrániť zo simulácie všetky inštancie nejakého typu cesty) alebo po vygenerovaní scenáru pomocou množstva “.xml” súborov. V týchto súboroch sa dá upraviť prakticky hocijaká časť simulácie napr.: “osm.passenger.trips” obsahuje informácie o všetkých vozidlách, časoch ich odchodu, ich rýchlosti a miesta ich štartu a destinácie. “osm.poly.xml” obsahuje informácie o okolí ciest: budovy, rieky, parky... “osm.net.xml” obsahuje informácie o cestách, ich type, povolených vozidlách... Všetky použité “.xml” súbory sú potom zapísané v “osm.sumocfg”, ktoré sa potom spúšťa v sumo-gui.

Úprava množstva áut a premávky pred vygenerovaním simulácie v OSM WW:



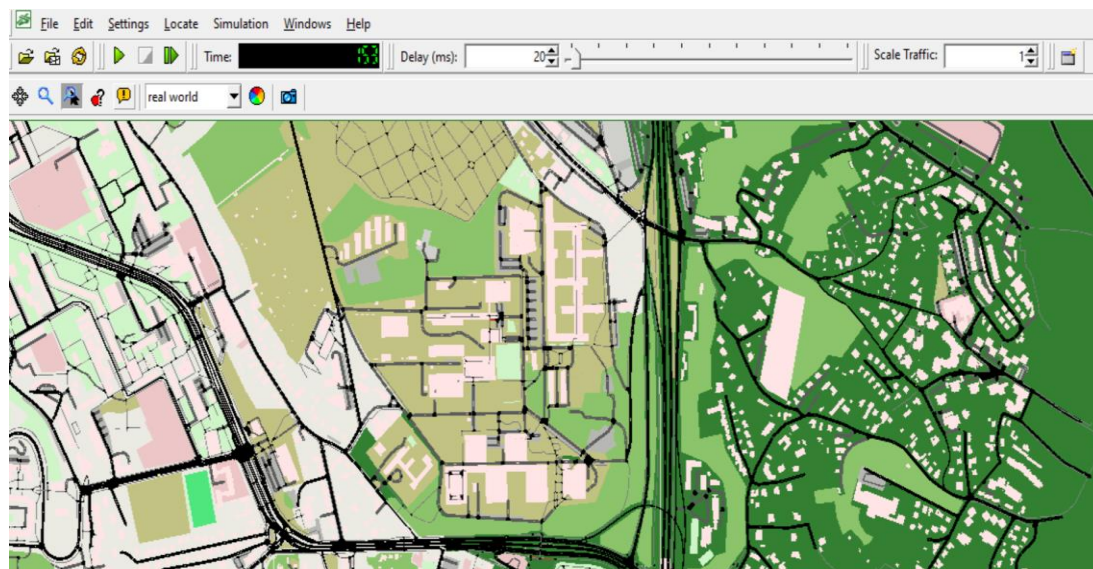
Obrázok 12 možnosť úpravy SUMO simulácie

Semaforey a cesty sa pravdepodobne najľahšie upravujú v NetEdit v GUI:



Obrázok 13 úprava existujúceho scenáru v NetEdit

V tejto práci som sa ďalej nezaoberal úpravami vygenerovaného scenáru. Prešiel som na iný taký, 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 predtým spomínaného súboru "osm.net.xml". Scenár pred úpravou s cestami pre bicykle (svetlo sivé cesty):



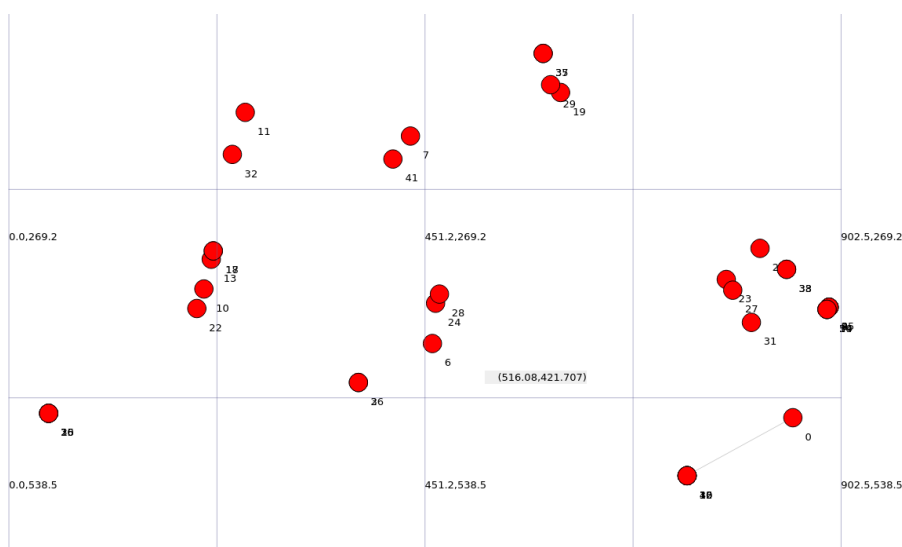
Obrázok 14 okolie FIIT STU v SUMO

Posledný scenár v SUMO, ktorý sa vytváral bol rovný úsek cesty R7 pri Slovnafte v Bratislave. Po vyskúšaní základných funkcionalít SUMO, bolo treba vyskúšať ns-3 a ns-3 + SUMO na jednoduchom rovnom úseku. Cieľom tu bolo na rovnom úseku otestovať komunikáciu medzi

autami (V2V) a vysielacími 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). Tu sa testovala point-to-point komunikácia, vytvárali sa uzly, ktoré predstavovali autá, vytvárali sa UDP echo server a klient, teda si medzi sebou zariadenia posielali UDP pakety. 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. Z “osm.sumocfg” sa pomocou SUMO vygeneruje “.xml”, ktorý obsahuje informácie o uzloch a ich súradniciach v čase. Z tohto “.xml” treba pomocou traceExporter.py vyrobiť “.tcl” súbor, ktorému rozumie ns-3. Do “mobility file” sa potom 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, tak ako na obrázku (v SUMO sa tam nenachádzali skoro žiadne autá):



Obrázok 15 ukážka uzlov pri automatickom generovaní

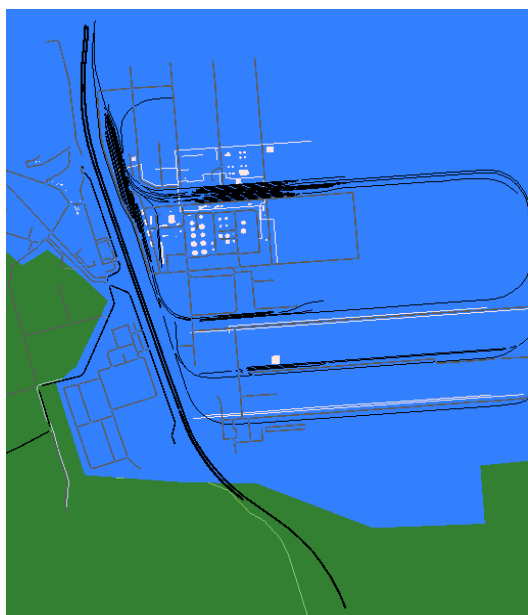
Preto som sa nakoniec rozhodol radšej v SUMO vypnúť automatické generovanie premávky (aj tak sa s ním ťažko narába, keďže sa štartovacie pozície/čas pre autá generujú náhodne a ich množstvo je určené počtom na kilometer jazdných pruhov) pre jednoduchosť som si teda manuálne vyklikal 4 autá, ktorým boli ako počiatočné a konečné hrany nastavené okraje diaľnice a tiež som do vygenerovaného “.tcl” súboru manuálne vložil uzly, ktoré predstavovali vysieláče popri diaľnici.

Ukážka upraveného “.tcl” súboru:

```
1 $node_(0) set X_ 1660.00
2 $node_(0) set Y_ 420.00
3 $node_(0) set Z_ 0
4 $node_(1) set X_ 1660.00
5 $node_(1) set Y_ 420.00
6 $node_(1) set Z_ 0
7 $node_(2) set X_ 2349.82
8 $node_(2) set Y_ 235.49
9 $node_(2) set Z_ 0
10 $node_(3) set X_ 1972.47
11 $node_(3) set Y_ 526.08
12 $node_(3) set Z_ 0
13 $node_(4) set X_ 1573.05
14 $node_(4) set Y_ 971.85
15 $node_(4) set Z_ 0
16 $node_(5) set X_ 1008.69
17 $node_(5) set Y_ 3365.24
18 $node_(5) set Z_ 0
```

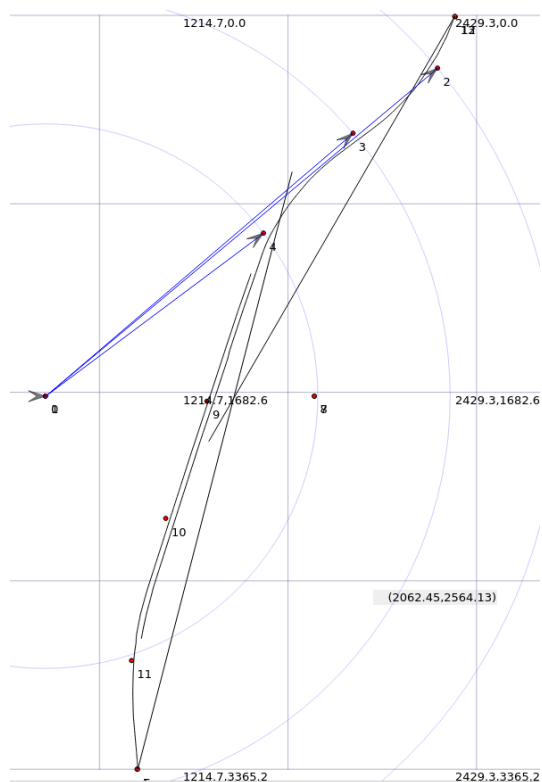
Obrázok 16 ukážka tcl súboru

Ďalej som sa pokúsil do NetAnim vložiť pozadový obrázok zo SUMO pre lepšiu ilustráciu simulácie. Toto tak úplne nefungovalo, keďže po vložení sa obrázok zo SUMO nezhodoval s pohybmi áut. Súradnice v NetAnim sú prehodené oproti SUMO a treba ich externe upraviť. Sledovaný úsek R7 v Bratislave v SUMO:

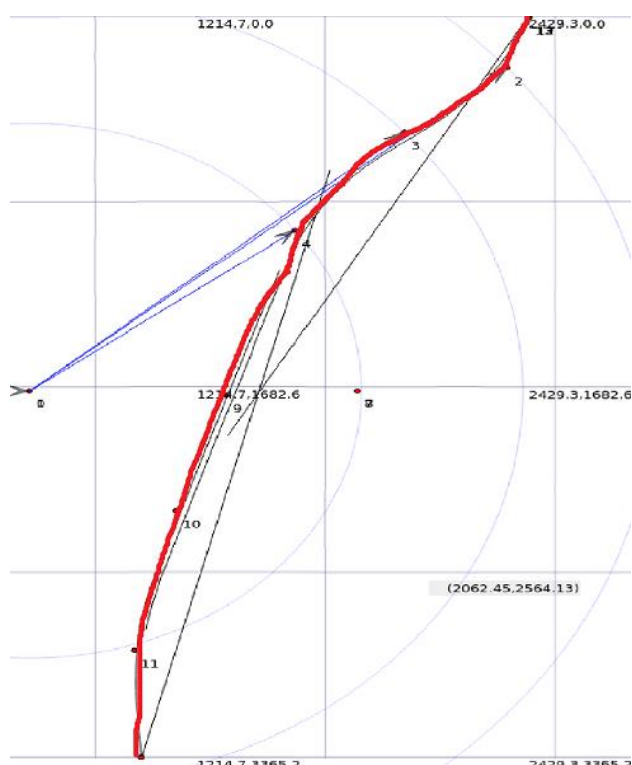


Obrázok 17 SUMO R7 v Bratislave

Trajektória pohyblivých uzlov (áut) na diaľnici:



Obrázok 18 trajektória pohybu vozidiel 1



Obrázok 19 trajektórie pohybu vozidiel 2

Z obrázkov je vidieť, že ten istý scenár je v ns-3 (NetAnim) zrkadlovo obrátený oproti SUMO podľa y osi. Je to kvôli tomu, že y súradnica rastie v ns-3 smerom nadol, ale v SUMO rastie

nahor. Keďže medzi SUMO a NetAnim nie je nejaký prevod súradníc, dochádza k situácii ako na obrázku. V tomto štádiu sa mi síce podarilo otestovať vloženie obrázku do NetAnim (funguje), ale bez zrkadlového obrátenia obrázku je aj tak nepoužiteľný, keďže nekopíruje trasu idúcich áut.

V ďalšom kroku som si tvoril vlastný scenár ns-3. Tu som vyskúšal použitie viacerých druhov fyzickej vrstvy (CSMA, wifi, point-to-point). Taktiež som testoval použitie viacerých podsietí. 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. Každé auto bolo UDP echo server a AP body boli klienti. 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” [26], 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 požiadavkách. Tento spôsob predávania si informácii medzi autami síce fungoval, ale potreboval zbytočné množstvo nastavovania a podsietí.

V nasledujúcom 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.” [46] 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 sa tu 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 sa už rozhodol prejsť s mojej verzie WSL, používal som WSL 1, ktorá neobsahovala linux kernel. [43] To bol problém keď som chcel použiť linux lxc kontajner na simuláciu reálneho zariadenia na použitie v ns-3 simulácii. [47] 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. Ale WSL2 má aj tak viacero výhod a upgrade je jednoduchý. Akurát po prejdení z WSL1 na WSL2 treba pri spúšťaní display serveru vypnúť firewall, inak sa nespustí žiadne GUI okno.

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” (aj súpravou tcl súboru), čo je pre

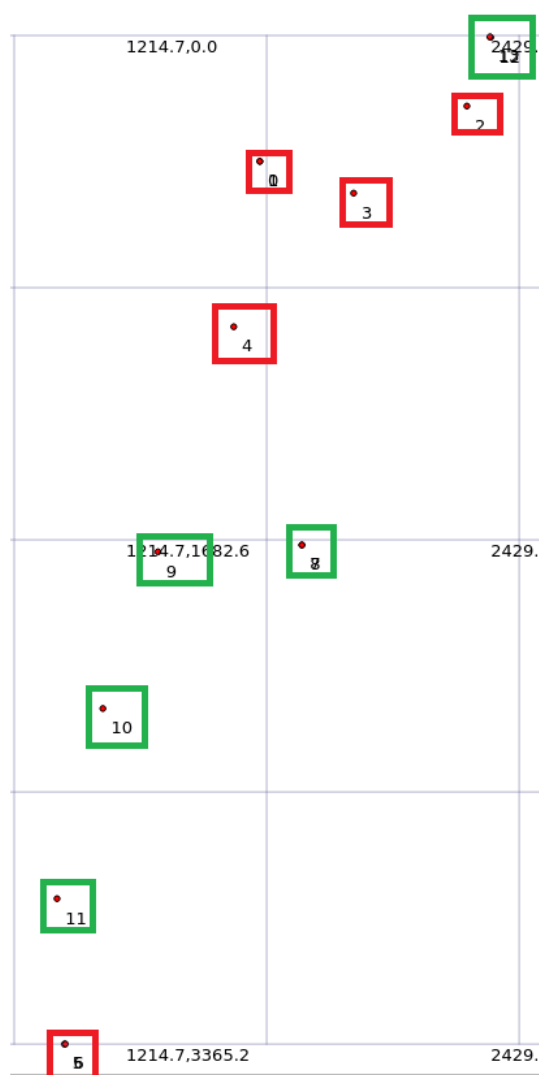
Ako posledný krok bolo rozdelenie všetkých vysielateľov, áut, satelitov do 2 skupín, ktoré medzi sebou nie sú schopné komunikovať. Napríklad na simulovanie zariadení, ktoré používajú 2G vs 3G... [49] A následná komunikácia zo simulácie pozostávala z vygenerovaných UDP paketov na jednom z áut, ktoré sa posielali na Linux rozhranie.

[illegible]

35

Na obrázku je ukážka finálnej podoby mojej simulácie v NetAnim. Červenou farbou sú zobrazené vysielacie pri diaľnici, vždy dvojica uzlov kde jeden je ghost node a mostom prepojený s TAP zariadením a druhý je nastavený ako gateway a komunikuje s ostatnými uzlami. Tieto slúžia ako AP body pre vysielacie pri diaľnici, ktoré sú v tomto prípade wireless STA. Zelenou farbou sú označené pohybujúce sa vozidlá, ktoré používajú ad-hoc na komunikáciu medzi sebou a inak v podsieti spolu s vysielacími sú STA zariadenia a vysielacie sú AP.

Ďalší obrázok ukazuje rozdelenie uzlov do jednotlivých podsietí:



Obrázok 21 NetAnim finálna verzia podsiete

Uzly sú logicky rozdelené do dvoch celkov (farebne oddelených), ktoré nie sú schopné medzi sebou komunikovať. Uzly s prekrytým číslovaním sú červené vozidlá 5 a 6, zelené vozidlá 12 a 13, červený ghost node a AP 0 a 1, zelený ghost node a AP 7 a 8.

Scenár obsahuje podsiete, tieto sú nastaviteľné, ale pre účely testovania budú takéto:

10.1.1.0 – uzly 0, 1, 2, 3, 4

10.1.2.0 – uzly 6, 3

10.1.3.0 – uzly 5, 6

10.1.4.0 – uzly 7, 8, 9, 10, 11

10.1.5.0 – uzly 10, 13

10.1.6.0 – uzly 12, 13

3.4 Overenie riešenia

3.4.1 Scenár 1

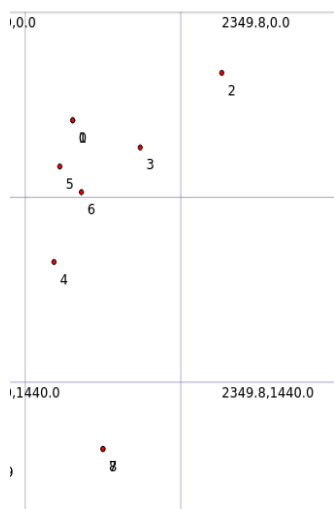
Požiadavka/y

Celú simuláciu by malo byť možné sledovať na GUI.

Popis testu

Po prebehnutí simulácie sa bude v NetAnim sledovať pohyb vozidiel a rozmiestnenie zariadení, možnosti úpravy simulácie.

Výstup



5	
Property	Value
Node Id	5
Node System Id	0
Node Description	5
▼ Node Position	
Node X	1,600.00
Node Y	600.00
▼ Node Color	■ [255, 0, 0] (255)
Red	255
Green	0
Blue	0
Alpha	255
Node Size	20.00
Node Resource	
Show Node Trajectory	<input type="checkbox"/> False
▼ Ipv4 Addresses	
127.0.0.1	
10.1.7.1	
10.1.3.1	
▼ Ipv6 Addresses	
::1	
▼ Mac Addresses	
00:00:00:00:00:06	

Obrázok 22 NetAnim úprava simulácie

V NetAnim je možné sledovať rozmiestnenie všetkých zariadení, ich súradníc a označenia. Ďalej je možné simuláciu zastaviť, vrátiť... Je tu možnosť zistiť MAC a IP adresy zariadení kliknutím na ich príslušný uzol. Dá sa tu pozrieť trajektória mobilných uzlov a tiež sa dá zmeniť farba uzlov.

Záver

NetAnim poskytuje celkom intuitívne GUI, ktoré slúži na zobrazenie simulácii.

3.4.2 Scenár 2

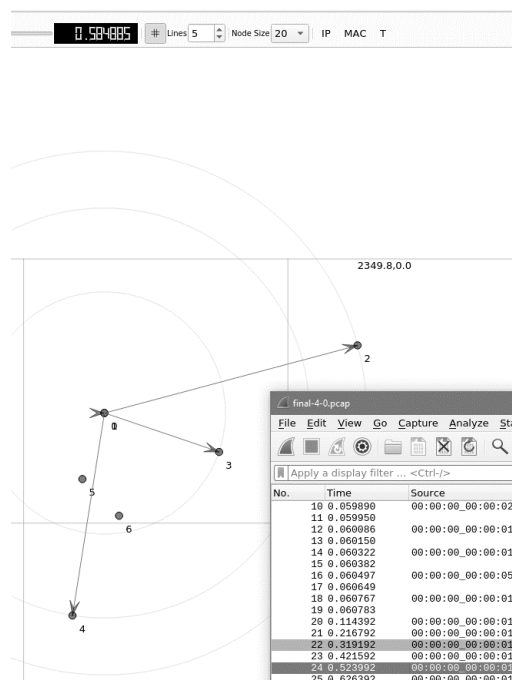
Požiadavka/y

Animovanie komunikácie medzi danými zariadeniami. Malo by byť vidieť sieťovú premávku medzi jednotlivými zariadeniami.

Popis testu

Po prebehnutí simulácie budem v NetAnim sledovať grafické znázornenie komunikácie medzi uzlami a kontrolovať jeho správnosť.

Výstup



Obrázok 23 NetAnim sledovanie animácie

Pri sledovaní simulácie v NetAnim, modré šípky znázorňujú poslanie paketu medzi uzlami a smer poslania tohto paketu. Podľa času simulácie (ľavý horný roh obrázku) a zachytenej

komunikácii vo Wireshark je vidieť, že čas poslania paketu podľa Wireshark a jeho grafického znázornenia podľa NetAnim sa nezhoduje a je zhruba o 0.06 sekundy posunutý. To môže súvisieť s faktom, že ns-3 začína s Wireshark capture až po prijatí prvého paketu na rozhraní, príklad v testovacom scenári 7.

Záver

NetAnim je schopný animovať komunikáciu medzi zariadeniami. Je vidieť graficky znázornené pakety, ktorých čas poslania nie je vždy rovnaký ako ich reálny čas poslania.

3.4.3 Scenár 3

Požiadavka/y

Sieťový simulátor by si mal vedieť vytvoriť vlastné rozhranie na skutočnom zariadení.

Sieťový simulátor by mal vedieť po konci simulácie vytvorené rozhranie zničiť.

Popis testu

Pomocou príkazu `ifconfig` sa bude sledovať, či sa ns-3 úspešne podarilo vytvoriť TAP rozhranie so zadanými parametrami.

Výstup

```
thetap: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500
    inet 10.1.1.1 netmask 255.255.255.0 broadcast 10.1.1.255
    inet6 fe80::200:ff:fe00:1 prefixlen 64 scopeid 0x20<link>
    ether 00:00:00:00:00:01 txqueuelen 1000 (Ethernet)
    RX packets 0 bytes 0 (0.0 B)
    RX errors 0 dropped 0 overruns 0 frame 0
    TX packets 3 bytes 266 (266.0 B)
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

thetap2: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500
    inet 10.1.4.1 netmask 255.255.255.0 broadcast 10.1.4.255
    inet6 fe80::200:ff:fe00:a prefixlen 64 scopeid 0x20<link>
    ether 00:00:00:00:00:0a txqueuelen 1000 (Ethernet)
    RX packets 0 bytes 0 (0.0 B)
    RX errors 0 dropped 0 overruns 0 frame 0
    TX packets 3 bytes 266 (266.0 B)
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

jakub@LAPTOP-10FH8F54:~/Desktop$
```

Obrázok 24 TAP rozhrania

Ns-3 vytvoril dve TAP rozhrania “thetap” a “thetap2” na Linux host zariadení. Tieto po skončení simulácie zmiznú.

Záver

Sieťový simulátor ns-3 dokáže vytvoriť vlastné TAP rozhranie/a, ktoré sa menom, IP a maskou zhodujú s hodnotami predtým zadávanými v simulácii. Po skončení simulácie ns-3 zavolá delete na tieto rozhrania.

3.4.4 Scenár 4

Požiadavka/y

Mal by sa dať nastaviť uzol, ktorý bude schopný generovať pakety.

Mal by sa dať nastaviť uzol, ktorý bude prijímať vygenerované pakety.

Zariadenia v simulácii by mali byť schopné posielat' pakety von na rozhranie môjho zariadenia.

Popis testu

UDP pakety sa budú posielat' z uzlu 13 (10.1.5.2) na rozhranie "thetap2" port 9. Tu ich budem zachytávať wireshark. Vozidlá sa pri tomto scenári nehýbu a sú stále na dosah.

```
Ptr<Node> appSource = NodeList::GetNode (13);
OnOffHelper onoff ("ns3::UdpSocketFactory", Address (InetSocketAddress ("10.1.4.1", 9))); //creates udp packets and sends them to 10.1.4.1, port 9
ApplicationContainer app = appFS_TestAll (appSource);
```

Obrázok 25 UDP generátor

Výstup

448	6.744549373	10.1.5.2	10.1.4.1	UDP	554 49153 → 9 Len=512
449	6.754111209	10.1.5.2	10.1.4.1	UDP	554 49153 → 9 Len=512
450	6.765773342	10.1.5.2	10.1.4.1	UDP	554 49153 → 9 Len=512
451	6.777595510	10.1.5.2	10.1.4.1	UDP	554 49153 → 9 Len=512
452	6.792847122	10.1.5.2	10.1.4.1	UDP	554 49153 → 9 Len=512
453	6.807072478	10.1.5.2	10.1.4.1	UDP	554 49153 → 9 Len=512
454	6.815423530	10.1.5.2	10.1.4.1	UDP	554 49153 → 9 Len=512
455	6.824044859	10.1.5.2	10.1.4.1	UDP	554 49153 → 9 Len=512
456	6.831886347	10.1.5.2	10.1.4.1	UDP	554 49153 → 9 Len=512
457	6.840314065	10.1.5.2	10.1.4.1	UDP	554 49153 → 9 Len=512

Wireshark · Packet 456 · thetap2	
<ul style="list-style-type: none"> Frame 456: 554 bytes on wire (4432 bits), 554 bytes captured (4432 bits) on interface thetap2, id 0 Ethernet II, Src: 00:00:00_00:00:0d (00:00:00:00:00:0d), Dst: 00:00:00_00:00:0a (00:00:00:00:00:0a) Internet Protocol Version 4, Src: 10.1.5.2, Dst: 10.1.4.1 User Datagram Protocol, Src Port: 49153, Dst Port: 9 Data (512 bytes) 	

Obrázok 26 UDP generátor Wireshark

Na obrázku je záznam z wireshark z rozhrania "thetap2". Je vidieť množstvo prichádzajúcich UDP paketov z 10.1.5.2 na 10.1.4.1 port 9. Za zmienku stojí, že sa tieto UDP pakety podarilo zachytiť aj na uzle 11 (10.1.4.5) cez ktorý nemali prechádzať.

Záver

Ns-3 má možnosť nastaviť uzol generujúci pakety, ďalej je možné nastaviť uzol prijímajúci pakety aj s portom a nakoniec tento uzol môže byť aj rozhranie na Linux host.

3.4.5 Scenár 5

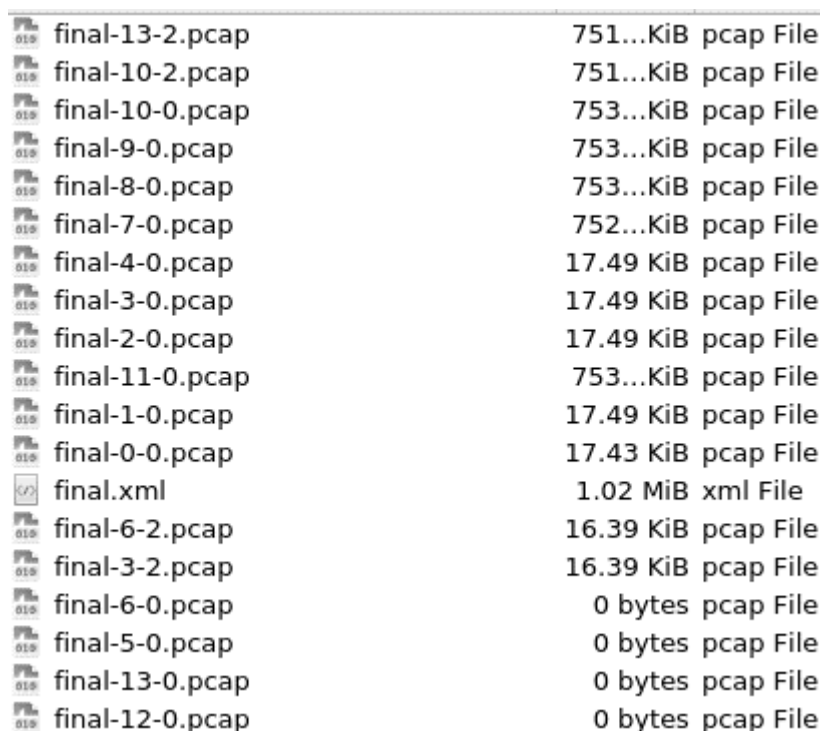
Požiadavka/y

Mal by som byť schopný zachytávať sieťovú premávku na vybraných zariadeniach a ich rozhraniach.

Popis testu

Po prebehnutí simulácie sa skontroluje množstvo vygenerovaných a názvy “.pcap” súborov.

Výstup



final-13-2.pcap	751...KiB	pcap File
final-10-2.pcap	751...KiB	pcap File
final-10-0.pcap	753...KiB	pcap File
final-9-0.pcap	753...KiB	pcap File
final-8-0.pcap	753...KiB	pcap File
final-7-0.pcap	752...KiB	pcap File
final-4-0.pcap	17.49 KiB	pcap File
final-3-0.pcap	17.49 KiB	pcap File
final-2-0.pcap	17.49 KiB	pcap File
final-11-0.pcap	753...KiB	pcap File
final-1-0.pcap	17.49 KiB	pcap File
final-0-0.pcap	17.43 KiB	pcap File
final.xml	1.02 MiB	xml File
final-6-2.pcap	16.39 KiB	pcap File
final-3-2.pcap	16.39 KiB	pcap File
final-6-0.pcap	0 bytes	pcap File
final-5-0.pcap	0 bytes	pcap File
final-13-0.pcap	0 bytes	pcap File
final-12-0.pcap	0 bytes	pcap File

Obrázok 27 vygenerované pcap súbory

Ns-3 počas simulácie generuje .pcap súbor (pomenovaný číslom uzla) pre každý uzol a každé rozhranie na danom uzle, preto sú tam niektoré dvakrát.

Záver

Ns-3 dokáže “zachytiť” premávku na jednotlivých zariadeniach a rozhraniach v podobe “.pcap” súborov.

3.4.6 Scenár 6

Požiadavka/y

Moje zariadenie by malo byť schopné posielat' pakety zariadeniam v simulácii.

Popis testu

Po vytvorení TAP sa pokúsim z Linux terminálu pingnúť uzol v simulácii. Vozidlá sa pri tomto scenári nehýbu a sú stále na dosah.

Výstup

```
jakub@LAPTOP-10FH8F54:~/ns-allinone-3.34/ns-3.34$ sudo route add -net 10.1.6.0 netmas
k 255.255.255.0 dev thetap2 gw 10.1.4.2
jakub@LAPTOP-10FH8F54:~/ns-allinone-3.34/ns-3.34$ ping 10.1.6.1
PING 10.1.6.1 (10.1.6.1) 56(84) bytes of data.
64 bytes from 10.1.6.1: icmp_seq=1 ttl=62 time=123 ms
64 bytes from 10.1.6.1: icmp_seq=2 ttl=62 time=34.2 ms
64 bytes from 10.1.6.1: icmp_seq=3 ttl=62 time=35.6 ms
64 bytes from 10.1.6.1: icmp_seq=4 ttl=62 time=29.5 ms
64 bytes from 10.1.6.1: icmp_seq=5 ttl=62 time=29.7 ms
64 bytes from 10.1.6.1: icmp_seq=6 ttl=62 time=28.7 ms

^C--- 10.1.6.1 ping statistics ---
6 packets transmitted, 6 received, 0% packet loss, time 5008ms
rtt min/avg/max/mdev = 28.675/46.867/123.452/34.345 ms
```

Obrázok 28 ping 1

No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000	00:00:00_00:00:10	Broadcast	ARP	64	Who has 10.1.6.1? Tell 10.1.6.2
2	0.000179	00:00:00_00:00:0f	00:00:00_00:00:10	ARP	64	10.1.6.1 is at 00:00:00:00:00:0f
3	0.000195		00:00:00_00:00:0f (...)	802.11	14	Acknowledgement, Flags=.....
4	0.000266	10.1.4.1	10.1.6.1	ICMP	120	Echo (ping) request id=0x24b6, seq=1/256, ttl=61 (reply in 9)
5	0.000350		00:00:00_00:00:10 (...)	802.11	14	Acknowledgement, Flags=.....
6	0.009452	00:00:00_00:00:0f	Broadcast	ARP	64	Who has 10.1.6.2? Tell 10.1.6.1
7	0.009486	00:00:00_00:00:10	00:00:00_00:00:0f	ARP	64	10.1.6.2 is at 00:00:00:00:00:10
8	0.009563		00:00:00_00:00:10 (...)	802.11	14	Acknowledgement, Flags=.....
9	0.009763	10.1.6.1	10.1.4.1	ICMP	120	Echo (ping) reply id=0x24b6, seq=1/256, ttl=64 (request in...
10	0.009779		00:00:00_00:00:0f (...)	802.11	14	Acknowledgement, Flags=.....
11	0.971512	10.1.4.1	10.1.6.1	ICMP	120	Echo (ping) request id=0x24b6, seq=2/512, ttl=61 (reply in 1...
12	0.971596		00:00:00_00:00:10 (...)	802.11	14	Acknowledgement, Flags=.....

Obrázok 29 Wireshark komunikácia 1

Na prvom obrázku je vidieť, že komunikácia medzi linux hostom s gateway 10.1.4.2 a uzlom 13 (10.1.6.1) prebehla úspešne. Nedošlo k žiadnej strate paketov. Tiež je vo wireshark možné sledovať ARP a ICMP komunikáciu medzi 10.1.4.1 (TAP rozhranie) a 10.1.6.1.

Záver

Bol som schopný z Linux hosta s použitím TAP rozhrania pingnúť uzol v ns-3 simulácii.

3.4.7 Scenár 7

Požiadavka/y

Sieťový simulátor by mal byť schopný komunikovať s niekoľkými reálnymi rozhraniami naraz.

Popis testu

Tu sa spustí kombinovaná simulácia. O ping sa pokúsim z dvoch TAP rozhraní na 2 rôzne uzly a zároveň budem posielat' UDP premávku na jedno z rozhraní.

Výstup

```
jakub@LAPTOP-10FH8F54:~/ns-allinone-3.34/ns-3.34$ sudo route add -net 10.1.3.0 netmask 255.255.255.0 dev thetap gw 10.1.1.2
jakub@LAPTOP-10FH8F54:~/ns-allinone-3.34/ns-3.34$ sudo route add -net 10.1.6.0 netmask 255.255.255.0 dev thetap2 gw 10.1.4.2
jakub@LAPTOP-10FH8F54:~/ns-allinone-3.34/ns-3.34$ ping 10.1.6.1
PING 10.1.6.1 (10.1.6.1) 56(84) bytes of data:
64 bytes from 10.1.6.1: icmp_seq=1 ttl=62 time=133 ms
64 bytes from 10.1.6.1: icmp_seq=2 ttl=62 time=154 ms
64 bytes from 10.1.6.1: icmp_seq=3 ttl=62 time=46.3 ms
^C
--- 10.1.6.1 ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 2003ms
rtt min/avg/max/mdev = 46.309/111.299/154.337/46.754 ms
jakub@LAPTOP-10FH8F54:~/ns-allinone-3.34/ns-3.34$ ping 10.1.3.1
PING 10.1.3.1 (10.1.3.1) 56(84) bytes of data:
64 bytes from 10.1.3.1: icmp_seq=1 ttl=62 time=438 ms
64 bytes from 10.1.3.1: icmp_seq=2 ttl=62 time=33.7 ms
64 bytes from 10.1.3.1: icmp_seq=3 ttl=62 time=63.2 ms
^C
--- 10.1.3.1 ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 2003ms
rtt min/avg/max/mdev = 33.682/178.207/437.789/183.945 ms
jakub@LAPTOP-10FH8F54:~/ns-allinone-3.34/ns-3.34$
```

Obrázok 30 ping 2

No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000	00:00:00_00:00:10	Broadcast	ARP	64	Who has 10.1.6.1? Tell 10.1.6.2
2	0.000179	00:00:00_00:00:0f	00:00:00_00:00:10	ARP	64	10.1.6.1 is at 00:00:00:00:00:0f
3	0.000195	00:00:00_00:00:0f	00:00:00_00:00:0f (...)	802.11	14	Acknowledgement, Flags=.....
4	0.000266	10.1.4.1	10.1.6.1	ICMP	120	Echo (ping) request id=0x274a, seq=1/256, ttl=61 (reply in 9)
5	0.000350	00:00:00_00:00:10	00:00:00_00:00:10 (...)	802.11	14	Acknowledgement, Flags=.....
6	0.009452	00:00:00_00:00:0f	Broadcast	ARP	64	Who has 10.1.6.2? Tell 10.1.6.1
7	0.009486	00:00:00_00:00:10	00:00:00_00:00:0f	ARP	64	10.1.6.2 is at 00:00:00:00:00:10
8	0.009563	00:00:00_00:00:10	00:00:00_00:00:10 (...)	802.11	14	Acknowledgement, Flags=.....
9	0.009763	10.1.6.1	10.1.4.1	ICMP	120	Echo (ping) reply id=0x274a, seq=1/256, ttl=64 (request in...)
10	0.009779	00:00:00_00:00:0f	00:00:00_00:00:0f (...)	802.11	14	Acknowledgement, Flags=.....

Obrázok 31 Wireshark komunikácia 2

No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000	00:00:00_00:00:07	Broadcast	ARP	64	Who has 10.1.3.1? Tell 10.1.3.2
2	0.000034	00:00:00_00:00:06	00:00:00_00:00:07	ARP	64	10.1.3.1 is at 00:00:00:00:00:06
3	0.000110	00:00:00_00:00:06	00:00:00_00:00:06 (...)	802.11	14	Acknowledgement, Flags=.....
4	0.000274	10.1.1.1	10.1.3.1	ICMP	120	Echo (ping) request id=0x2750, seq=1/256, ttl=61 (reply in 9)
5	0.000290	00:00:00_00:00:07	00:00:00_00:00:07 (...)	802.11	14	Acknowledgement, Flags=.....
6	0.002308	00:00:00_00:00:06	Broadcast	ARP	64	Who has 10.1.3.2? Tell 10.1.3.1
7	0.002487	00:00:00_00:00:07	00:00:00_00:00:06	ARP	64	10.1.3.2 is at 00:00:00:00:00:07
8	0.002583	00:00:00_00:00:07	00:00:00_00:00:07 (...)	802.11	14	Acknowledgement, Flags=.....
9	0.002655	10.1.3.1	10.1.1.1	ICMP	120	Echo (ping) reply id=0x2750, seq=1/256, ttl=64 (request in...)
10	0.002740	00:00:00_00:00:06	00:00:00_00:00:06 (...)	802.11	14	Acknowledgement, Flags=.....

Obrázok 32 Wireshark komunikácia 3

82	3.979142	00:00:00_00:00:0d	Broadcast	ARP	64	Who has 10.1.4.1? Tell 10.1.4.4
83	3.979158	00:00:00_00:00:0d	00:00:00_00:00:0d (...)	802.11	14	Acknowledgement, Flags=.....
84	3.979236	00:00:00_00:00:0d	Broadcast	ARP	64	Who has 10.1.4.1? Tell 10.1.4.4
85	3.992951	00:00:00_00:00:0a	00:00:00_00:00:0d	ARP	64	10.1.4.1 is at 00:00:00:00:00:0a
86	3.993128	00:00:00_00:00:0a	00:00:00_00:00:0a (...)	802.11	14	Acknowledgement, Flags=.....
87	3.993954	10.1.5.2	10.1.4.1	UDP	576	49153 → 9 Len=512
88	3.993970	00:00:00_00:00:0d	00:00:00_00:00:0d (...)	802.11	14	Acknowledgement, Flags=.....
89	3.994846	10.1.5.2	10.1.4.1	UDP	576	49153 → 9 Len=512
90	3.994862	00:00:00_00:00:0d	00:00:00_00:00:0d (...)	802.11	14	Acknowledgement, Flags=.....

Obrázok 33 Wireshark komunikácia 4

Na vrchnom obrázku je vidieť úspešné ping-y z obidvoch TAP rozhraní na vozidlá v ich časti simulácie. Ďalej je vidieť ICMP a ARP komunikáciu pre thetap (obr.1) a thetap2 (obr.2) a nakoniec aj vygenerované UDP pakety zachytené na thetap2. Za povšimnutie stojí aj fakt, že

prvý ARP request bol v oboch prípadoch poslaný v čase 0 (aj keď ping request-y boli poslané v rôznyi čas). Ns-3 spúšťa wireshark až po prijatí prvého paketu na rozhraní.

Záver

Ns-3 je schopné zniest' viacero komunikujúcich TAP rozhraní v jednej simulácii (vhodné na prepájanie viacerých skutočných rozhraní cez ns-3 simuláciu).

3.4.8 Scenár 8

Požiadavka/y

Sieťový simulátor by mal podporovať rôzne druhy routovacích protokolov.

Popis testu

Namiesto globálneho routovania bude použitý OLSR routovací protokol. Následne bude vo wireshark skontrolované prebiehanie OLSR paketov.

Výstup

olsr						
No.	Time	Source	Destination	Protocol	Length	Info
363	20.397105	10.1.1.2	10.1.1.255	OLSR v1	120	OLSR (IPv4) Packet, Length: 56 Bytes
365	20.397952	10.1.1.4	10.1.1.255	OLSR v1	212	OLSR (IPv4) Packet, Length: 148 Bytes
367	20.398568	10.1.1.4	10.1.1.255	OLSR v1	212	OLSR (IPv4) Packet, Length: 148 Bytes
384	22.018784	10.1.1.1	10.1.1.255	OLSR v1	84	OLSR (IPv4) Packet, Length: 20 Bytes
385	22.050033	10.1.1.3	10.1.1.255	OLSR v1	120	OLSR (IPv4) Packet, Length: 56 Bytes
387	22.050312	10.1.1.3	10.1.1.255	OLSR v1	120	OLSR (IPv4) Packet, Length: 56 Bytes
390	22.276478	10.1.1.5	10.1.1.255	OLSR v1	120	OLSR (IPv4) Packet, Length: 56 Bytes
392	22.276757	10.1.1.5	10.1.1.255	OLSR v1	120	OLSR (IPv4) Packet, Length: 56 Bytes
393	22.341496	10.1.1.2	10.1.1.255	OLSR v1	120	OLSR (IPv4) Packet, Length: 56 Bytes

Obrázok 34 Wireshark OLSR capture

Na obrázku je vidieť, že každý uzol posiela vlastné OLSR pakety.

▶ Frame 423: 120 bytes on wire (960 bits), 120 bytes captured (960 bits)
▶ IEEE 802.11 Data, Flags:T
▶ Logical-Link Control
▶ Internet Protocol Version 4, Src: 10.1.1.5, Dst: 10.1.1.255
▶ User Datagram Protocol, Src Port: 698, Dst Port: 698
▼ Optimized Link State Routing Protocol
Packet Length: 56
Packet Sequence Number: 12
▶ Message: HELLO (1)

Obrázok 35 Wireshark OLSR správy

```

▶ Frame 448: 108 bytes on wire (864 bits), 108 bytes captured (864 bits)
▶ IEEE 802.11 Data, Flags: .....T
▶ Logical-Link Control
▶ Internet Protocol Version 4, Src: 10.1.1.4, Dst: 10.1.1.255
▶ User Datagram Protocol, Src Port: 698, Dst Port: 698
▼ Optimized Link State Routing Protocol
    Packet Length: 44
    Packet Sequence Number: 18
    ▶ Message: TC (2)
    ▶ Message: MID (3)

```

Obrázok 36 Wireshark OLSR správy 2

Na obrázkoch vyššie je vidieť aj OLSR správy teda: HELLO na nájdenie susedov na jeden skok a susedov na 2 skoky odpoveďou, MID obsahuje informácie o adresách na rozhraniach uzla, TC deklaruje topológiu. [50]

Záver

Ns-3 pozná rôzne druhy routovacích protokolov medzi nimi aj OLSR.

3.4.9 Scenár 9

Požiadavka/y

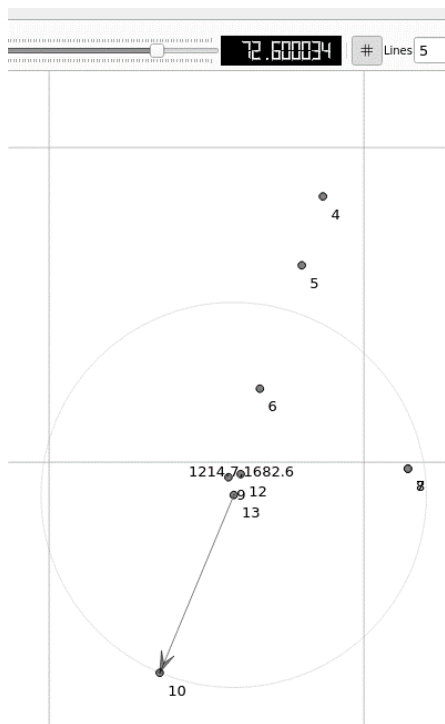
Moje zariadenie by malo byť schopné posilať pakety zariadeniam v simulácii.

Popis testu

Pokusím sa z Linux terminálu pingnúť uzol v simulácii. Tentoraz sa však vozidlá budú pohybovať. Z Linux host bude pingnuté auto - uzol 13. ICMP pakety teda traverzujú sieť takýmto spôsobom: TAP rozhranie (uzol 7 a "ghost node") -> vysielateľ (uzol 8 – STA zariadenie) -> vysielateľ (uzol 10 – AP zariadenie) -> auto (uzol 13 – STA zariadenie) -> auto (uzol 13 – Ad hoc) -> auto (uzol 12 - Ad hoc).

Výstup

Keďže kanál medzi vozidlom 12 a vysielateľom 10 používa LogDistancePropagationLossModel, vzdialenosť na ktorú dokážu komunikovať je obmedzená. Komunikácia teda neprebehne až pokiaľ sa vozidlá nepriblížia dostatočne blízko k vysielateľu. V NetAnim je na obrázku znázornený čas, kedy vysielateľ 10 a auto 13 začali komunikáciu:



Obrázok 37 NetAnim pohyblivé uzly animácia

Tento moment sa dá sledovať aj v termináli na Linux host:

```
jakub@LAPTOP-10FH8F54:~/Desktop$ ping 10.1.6.1
PING 10.1.6.1 (10.1.6.1) 56(84) bytes of data.
64 bytes from 10.1.6.1: icmp_seq=56 ttl=62 time=1051 ms
64 bytes from 10.1.6.1: icmp_seq=57 ttl=62 time=38.9 ms
64 bytes from 10.1.6.1: icmp_seq=58 ttl=62 time=28.9 ms
64 bytes from 10.1.6.1: icmp_seq=59 ttl=62 time=27.6 ms
64 bytes from 10.1.6.1: icmp_seq=60 ttl=62 time=38.0 ms
64 bytes from 10.1.6.1: icmp_seq=61 ttl=62 time=35.2 ms
```

Prvému ICMP reply to trvalo 1051ms, pokým sa auto priblížilo k vysielacu.

Vo wireshark na uzol 10:

765	73.202794	00:00:00_00:00:11	Broadcast	ARP	64 Who has 10.1.5.2? Tell 10.1.5.1
766	73.203056	00:00:00_00:00:12	00:00:00_00:00:12	ARP	64 10.1.5.2 is at 00:00:00:00:00:12
767	73.203072	00:00:00_00:00:12	...	802.11	14 Acknowledgement, Flags=.....
768	73.203213	10.1.4.1	10.1.6.1	ICMP	120 Echo (ping) request id=0x4907, seq=56/14336, ttl=62 (reply i...
769	73.203460	00:00:00_00:00:11	...	802.11	14 Acknowledgement, Flags=.....
770	73.216000	00:00:00_00:00:11	Broadcast	802.11	58 Beacon frame, SN=763, FN=0, Flags=....., BI=100, SSID=cent...
771	73.221822	00:00:00_00:00:12	Broadcast	ARP	64 Who has 10.1.5.1? Tell 10.1.5.2
772	73.221838	00:00:00_00:00:12	...	802.11	14 Acknowledgement, Flags=.....
773	73.221916	00:00:00_00:00:12	Broadcast	ARP	64 Who has 10.1.5.1? Tell 10.1.5.2
774	73.222152	00:00:00_00:00:12	00:00:00_00:00:12	ARP	64 10.1.5.1 is at 00:00:00:00:00:11
775	73.222327	00:00:00_00:00:11	...	802.11	14 Acknowledgement, Flags=.....
776	73.222581	10.1.6.1	10.1.4.1	ICMP	120 Echo (ping) reply id=0x4907, seq=56/14336, ttl=63 (request...
777	73.222587	00:00:00_00:00:12	...	802.11	14 Acknowledgement, Flags=.....

Obrázok 38 Wireshark komunikácia 5

Posielali sa len ARP requesty, nakoniec zhruba v 70. sekunde sa priblížil vysielac a auto a mohli si predávať ICMP správy.

Záver

Ns-3 je schopné si poradiť s posielaním reálnych paketov do simulácie aj vtedy keď nie je kvôli vzdialenosti uzlov možná ich okamžitá komunikácia. Pakety začne posielat' až keď sa dané uzly dostanú na dosah.

3.4.10 Scenár 10

Požiadavka/y

Mal by sa dať nastaviť uzol, ktorý bude schopný generovať pakety.

Mal by sa dať nastaviť uzol, ktorý bude prijímať vygenerované pakety.

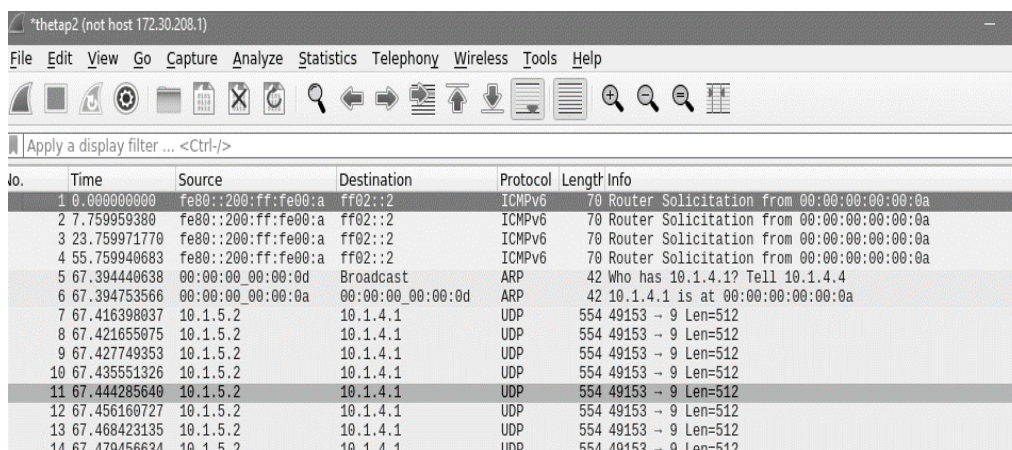
Zariadenia v simulácii by mali byť schopné posielat' pakety von na rozhranie môjho zariadenia.

Popis testu

Komunikácia von zo simulácie s globálnou smerovacou tabuľkou, pohyblivými uzlami a jedným AP/STA bodom. Simulácia je zameraná na posielanie UDP paketov zo simulácie von na tap rozhranie. V simulácii sa vytvárajú pakety na aute - uzle 13. Teda pakety traverzujú sieť rovnako ako v predošlom scenári, len na opačnú stranu a s vynechaním uzlu 12.

Výstup

Na thetap2 som sledoval premávku:



No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000000	fe80::200:ff:fe00:a	ff02::2	ICMPv6	70	Router Solicitation from 00:00:00:00:00:0a
2	7.759959380	fe80::200:ff:fe00:a	ff02::2	ICMPv6	70	Router Solicitation from 00:00:00:00:00:0a
3	23.759971770	fe80::200:ff:fe00:a	ff02::2	ICMPv6	70	Router Solicitation from 00:00:00:00:00:0a
4	55.759940683	fe80::200:ff:fe00:a	ff02::2	ICMPv6	70	Router Solicitation from 00:00:00:00:00:0a
5	67.394440638	00:00:00_00:00:0d	Broadcast	ARP	42	Who has 10.1.4.1? Tell 10.1.4.4
6	67.394753566	00:00:00_00:00:0a	00:00:00_00:00:0d	ARP	42	10.1.4.1 is at 00:00:00:00:00:0a
7	67.416398037	10.1.5.2	10.1.4.1	UDP	554	49153 → 9 Len=512
8	67.421655075	10.1.5.2	10.1.4.1	UDP	554	49153 → 9 Len=512
9	67.427749353	10.1.5.2	10.1.4.1	UDP	554	49153 → 9 Len=512
10	67.435551326	10.1.5.2	10.1.4.1	UDP	554	49153 → 9 Len=512
11	67.444285640	10.1.5.2	10.1.4.1	UDP	554	49153 → 9 Len=512
12	67.456160727	10.1.5.2	10.1.4.1	UDP	554	49153 → 9 Len=512
13	67.468423135	10.1.5.2	10.1.4.1	UDP	554	49153 → 9 Len=512
14	67.4704566634	10.1.5.2	10.1.4.1	UDP	554	49153 → 9 Len=512

Obrázok 39 Wireshark komunikácia 6

Žiadne pakety neprichádzali až do 67. sekundy kedy sa vozidlo 13 a vysielateľ 10 priblížili na dostatočnú vzdialenosť a začali komunikovať.

V Netanime je tento čas začiatku komunikácie zdanlivo oneskorený oproti wireshark, keďže wireshark capture nebol spustený zároveň so simuláciou, ale manuálne až po vytvorení TAP rozhrania.

Záver

Ns-3 je schopné si poradiť s posielaním reálnych paketov zo simulácie na TAP aj vtedy keď nie je kvôli vzdialenosti uzlov možná ich okamžitá komunikácia. Pakety začne posielat' až keď sa dané uzly dostanú na dosah.

3.4.11 Testy na otestovanie budúcich možností

Nasledujúca skupina testov slúži otestovanie budúcich možností simulácie.

3.4.12 Scenár 11

Cieľ

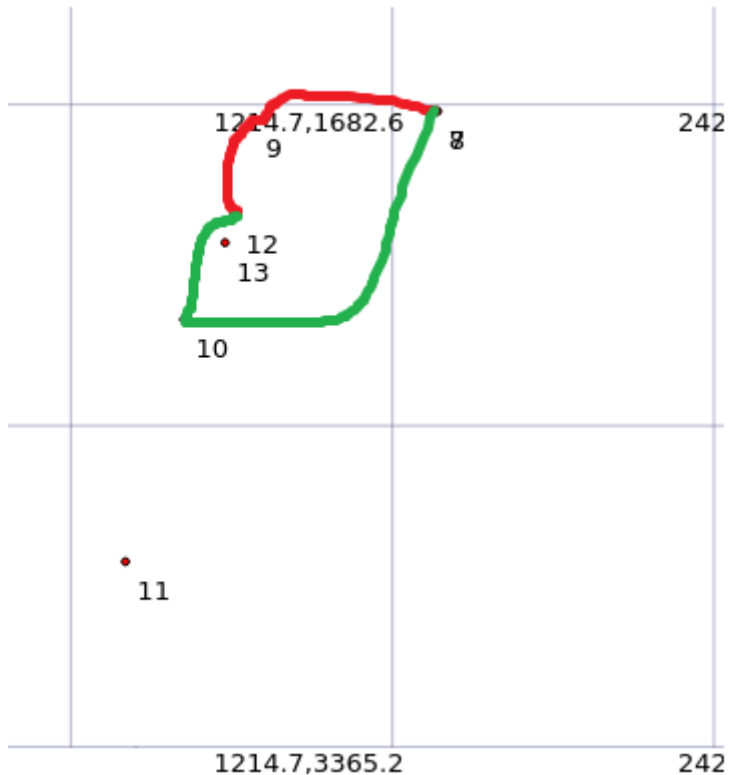
Zisti, či je možné z terminálu pingnúť akýkoľvek uzol, ktorý existuje v simulácii. Ak sú všetky vysielace a vozidlá v jednej podsieti.

Popis testu

Pokúsím sa pingnúť pohybujúce sa vozidlo 12. Tentoraz však sú všetky vysielace a autá v jednej podsieti a preto nemusia pakety traverzovať cez uzol 10 a 13 ako v predošlých scenároch.

Výstup

Po spustení ns-3 simulácia končí s error správou - msg="Assumed there is at most one exit from the root to this vertex", to znamená, že medzi TAP rozhraním a vozidlom 13 existuje viacero možných ciest s rovnakou cenou. Tu sú niektoré možné cesty:



Obrázok 40 NetAnim znázornenie routovacích ciest

Vyzerá, že si ns-3 nevie poradiť s niečím takýmto: červená cesta TAP 7 -> vysielateľ 9 -> auto 12 alebo zelená TAP 7 -> vysielateľ 10 -> auto 12.

Záver

Nie, s momentálnymi nastaveniami nie je možné simuláciu spustiť.

3.4.13 Scenár 12

Cieľ

Zisti, či je možné z terminálu pingnúť ľubovoľný uzol. S použitím OLSR smerovacieho protokolu.

Popis testu

Vozidlá sa nebudú pohybovať a pokúsim sa pingnúť uzol 12 pričom na routovanie bude použitý OLSR protokol.

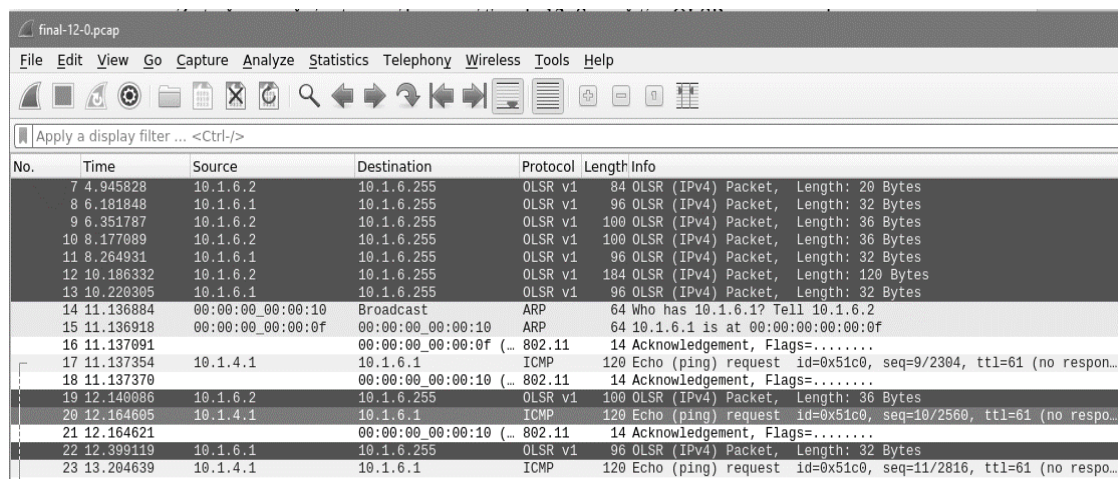
Výstup

Pokúsil som sa pingnúť vozidlo na uzle 12. Po spustení a pokuse o pingnutie sa stratilo 100% paketov:

```
jakub@LAPTOP-10FH8F54:~/Desktop$ ping 10.1.6.1
PING 10.1.6.1 (10.1.6.1) 56(84) bytes of data.
^C
--- 10.1.6.1 ping statistics ---
19 packets transmitted, 0 received, 100% packet loss, time 18744ms
```

Obrázok 41 ping neúspešný

Na uzle 12 je vidieť, že ICMP requesty sa dostali až sem, ale nikdy na ne nebola vygenerovaná odpoveď:



No.	Time	Source	Destination	Protocol	Length	Info
7	4.945828	10.1.6.2	10.1.6.255	OLSR v1	84	OLSR (IPv4) Packet, Length: 20 Bytes
8	6.481848	10.1.6.1	10.1.6.255	OLSR v1	96	OLSR (IPv4) Packet, Length: 32 Bytes
9	6.351787	10.1.6.2	10.1.6.255	OLSR v1	100	OLSR (IPv4) Packet, Length: 36 Bytes
10	8.177089	10.1.6.2	10.1.6.255	OLSR v1	100	OLSR (IPv4) Packet, Length: 36 Bytes
11	8.264931	10.1.6.1	10.1.6.255	OLSR v1	96	OLSR (IPv4) Packet, Length: 32 Bytes
12	10.186332	10.1.6.2	10.1.6.255	OLSR v1	184	OLSR (IPv4) Packet, Length: 120 Bytes
13	10.220305	10.1.6.1	10.1.6.255	OLSR v1	96	OLSR (IPv4) Packet, Length: 32 Bytes
14	11.136884	00:00:00_00:00:10	Broadcast	ARP	64	Who has 10.1.6.1? Tell 10.1.6.2
15	11.136918	00:00:00_00:00:0f	00:00:00_00:00:10	ARP	64	10.1.6.1 is at 00:00:00:00:00:0f
16	11.137091		00:00:00_00:00:0f (...)	802.11	14	Acknowledgement, Flags=.....
17	11.137354	10.1.4.1	10.1.6.1	ICMP	120	Echo (ping) request id=0x51c0, seq=9/2304, ttl=61 (no respon...
18	11.137370		00:00:00_00:00:10 (...)	802.11	14	Acknowledgement, Flags=.....
19	12.140086	10.1.6.2	10.1.6.255	OLSR v1	100	OLSR (IPv4) Packet, Length: 36 Bytes
20	12.164685	10.1.4.1	10.1.6.1	ICMP	120	Echo (ping) request id=0x51c0, seq=10/2560, ttl=61 (no respo...
21	12.164621		00:00:00_00:00:10 (...)	802.11	14	Acknowledgement, Flags=.....
22	12.399119	10.1.6.1	10.1.6.255	OLSR v1	96	OLSR (IPv4) Packet, Length: 32 Bytes
23	13.204639	10.1.4.1	10.1.6.1	ICMP	120	Echo (ping) request id=0x51c0, seq=11/2816, ttl=61 (no respo...

Obrázok 42 Wireshark komunikácia 7

Záver

Pri použití OLSR routovacieho protokolu nebolo možné dostať odpoveď na ICMP správy. Pakety s ICMP request sa síce dostali k požadovanému cieľu, ale odpoveď na ne neprišla.

3.4.14 Scenár 13

Cieľ

Zistiť, či je možné komunikovať z jedného TAP rozhrania na druhé TAP rozhranie cez simuláciu. Na tento účel existuje v mojej simulácii nastavenie, ktoré prepojí spoločnou podsietkou uzly 12 a 5.

Popis testu

Uzly sa nebudú pohybovať a pokúsim sa pingnúť z thetap2 na uzle 7 thetap na uzle 0.

Výstup

Vyzerá, že ping z thetap2 prešiel na thetap:

```
jakub@LAPTOP-10FH8F54:~/Desktop$ ping 10.1.1.1
PING 10.1.1.1 (10.1.1.1) 56(84) bytes of data.
64 bytes from 10.1.1.1: icmp_seq=1 ttl=64 time=0.743 ms
64 bytes from 10.1.1.1: icmp_seq=2 ttl=64 time=0.052 ms
64 bytes from 10.1.1.1: icmp_seq=3 ttl=64 time=0.050 ms
64 bytes from 10.1.1.1: icmp_seq=4 ttl=64 time=0.056 ms
64 bytes from 10.1.1.1: icmp_seq=5 ttl=64 time=0.045 ms
64 bytes from 10.1.1.1: icmp_seq=6 ttl=64 time=0.484 ms
64 bytes from 10.1.1.1: icmp_seq=7 ttl=64 time=0.054 ms
--
```

Obrázok 43 Wireshark ping 3

Avšak súdiac podľa času pingnutia a faktu, že som túto ICMP komunikáciu nevedel zachytiť na žiadnom uzle, som predpokladal, že komunikácia prebehla na loopbacku v rámci toho istého zariadenia bez prechodu ns-3 simuláciou.

Pri ďalšom spustení som zachytával premávku na loopback:

No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000000	10.1.1.1	10.1.1.1	ICMP	98	Echo (ping) request id=0x56c5, seq=1/256, ttl=64 (req
2	0.000607742	10.1.1.1	10.1.1.1	ICMP	98	Echo (ping) reply id=0x56c5, seq=1/256, ttl=64 (rec
3	1.017691412	10.1.1.1	10.1.1.1	ICMP	98	Echo (ping) request id=0x56c5, seq=2/512, ttl=64 (req
4	1.017703264	10.1.1.1	10.1.1.1	ICMP	98	Echo (ping) reply id=0x56c5, seq=2/512, ttl=64 (rec
5	2.057694539	10.1.1.1	10.1.1.1	ICMP	98	Echo (ping) request id=0x56c5, seq=3/768, ttl=64 (req
6	2.057705630	10.1.1.1	10.1.1.1	ICMP	98	Echo (ping) reply id=0x56c5, seq=3/768, ttl=64 (rec
7	3.097711676	10.1.1.1	10.1.1.1	ICMP	98	Echo (ping) request id=0x56c5, seq=4/1024, ttl=64 (re
8	3.097729470	10.1.1.1	10.1.1.1	ICMP	98	Echo (ping) reply id=0x56c5, seq=4/1024, ttl=64 (re
9	4.137714203	10.1.1.1	10.1.1.1	ICMP	98	Echo (ping) request id=0x56c5, seq=5/1280, ttl=64 (re
10	4.137725385	10.1.1.1	10.1.1.1	ICMP	98	Echo (ping) reply id=0x56c5, seq=5/1280, ttl=64 (re
11	5.177680377	10.1.1.1	10.1.1.1	ICMP	98	Echo (ping) request id=0x56c5, seq=6/1536, ttl=64 (re
12	5.177691819	10.1.1.1	10.1.1.1	ICMP	98	Echo (ping) reply id=0x56c5, seq=6/1536, ttl=64 (re
13	6.217713747	10.1.1.1	10.1.1.1	ICMP	98	Echo (ping) request id=0x56c5, seq=7/1792, ttl=64 (re

Obrázok 44 Wireshark komunikácia loopback

Teda skutočne komunikácia neprebehla cez ns-3 simuláciu.

Vyskúšal som ešte pingnúť uzol 1, teda posledný uzol pred ghost node pre TAP, pričom som použil ping -I, ktorý by mal prinútiť posielanie paketov cez zadané rozhranie teda thetap2 uzol 7:

```
jakub@LAPTOP-10FH8F54:~/Desktop$ sudo ping -I 10.1.4.1 10.1.1.2
PING 10.1.1.2 (10.1.1.2) from 10.1.4.1 : 56(84) bytes of data.
64 bytes from 10.1.1.2: icmp_seq=1 ttl=58 time=261 ms
64 bytes from 10.1.1.2: icmp_seq=2 ttl=58 time=76.0 ms
64 bytes from 10.1.1.2: icmp_seq=3 ttl=58 time=68.4 ms
64 bytes from 10.1.1.2: icmp_seq=4 ttl=58 time=76.8 ms
64 bytes from 10.1.1.2: icmp_seq=5 ttl=58 time=76.0 ms
```

Obrázok 45 Wireshark ping 4

196	15.184937	00:00:00_00:00:01	Broadcast	802.11	56 Beacon frame, SN=166, FN=0, Flags=....., BI=100, SSID=left
197	15.274375	10.1.4.1	10.1.1.2	ICMP	120 Echo (ping) request id=0x5812, seq=2/512, ttl=57 (no respons...
198	15.274435		00:00:00_00:00:04 (...)	802.11	14 Acknowledgement, Flags=.....
199	15.274653	10.1.4.1	10.1.1.2	ICMP	120 Echo (ping) request id=0x5812, seq=2/512, ttl=57 (reply in 2...
200	15.274669		00:00:00_00:00:01 (...)	802.11	14 Acknowledgement, Flags=.....
201	15.274747	10.1.1.2	10.1.4.1	ICMP	120 Echo (ping) reply id=0x5812, seq=2/512, ttl=64 (request in...
202	15.274991		00:00:00_00:00:02 (...)	802.11	14 Acknowledgement, Flags=.....
203	15.275281	10.1.1.2	10.1.4.1	ICMP	120 Echo (ping) reply id=0x5812, seq=2/512, ttl=64
204	15.275344		00:00:00_00:00:01 (...)	802.11	14 Acknowledgement, Flags=.....

Obrázok 46 Wireshark komunikácia 8

Podobne vyzerala zachytené premávka aj na uzle 7, teda sa mi podarilo vynútiť dlhšiu cestu cez thetap2. Nanešťastie táto technika nefungovala pre thetap, ktorý sa aj tak nepodarilo pingnúť cez simuláciu.

Záver

Pri súčasných nastaveniach nie je možné pingnúť jedno TAP rozhranie z druhého cez simuláciu. Je ale možné pingnúť párový uzol ku ghost node.

4 Zhodnotenie

Najprv som podľa tutoriálov a dokumentácie stiahol WSL1, ktorý som neskôr vylepšil na WSL2. Potom sa mi podarilo stiahnuť SUMO a ns-3. Pomocou SUMO dokumentácie som zistil možnosti modifikácie dopravnej simulácie. Ďalej som kontrolou s fotografiami z reálneho sveta objavil niekoľko problémov simulovania skutočných križovatiek v SUMO. Neskôr som podľa tutoriálov skombinoval vytvorené dopravné simulácie so sieťovým simulátorom ns-3, vytvorením súborov, ktoré obsahovali súradnice zariadení a ich pohyb. Sieťovú simuláciu som následne upravoval vďaka ns-3 dokumentácii a zisťoval vlastnosti ns-3. Nakoniec som umožnil komunikáciu môjho Linux host-a s ns-3 simuláciou s použitím TAP rozhraní. 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 s možnosťou úpravy hustoty premávky, rozmiestnenia ciest/budov, úpravou semaforov... 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 ns-3 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 posielat' pakety z reálneho zariadenia (WSL) do simulácie a tiež zo simulácie na reálne zariadenie.

4.1 Práca do budúcnosti

Práca do budúcnosti vychádza hlavne z posledných 3 testovacích scenárov.

Vyskúšať pomocou nastavení globálneho routovania zabezpečiť aby simulácia prebehla aj keď existuje niekoľko ciest medzi jednotlivými uzlami. Možno upraviť niektoré cesty aby stáli viac (toto by mohlo závisieť napríklad od vzdialenosti jednotlivých uzlov).

Ďalej som si všimol, že sa objavuje UDP/ICMP premávka aj na uzloch cez, ktoré by nemala ísť, ale sú v podsieti cez ktorú premávka ide. Treba zistiť prečo.

Otestovať prečo nefunguje ping reply pri použití OLSR protokolu, pravdepodobne bude treba podrobnejšie preskúmať olsr pakety. Možno má TAP zariadenie nejaký problém s predávaním svojich routovacích informácií.

Otestovať pri komunikácii z TAP na TAP cez ns-3 možnosť vymazania routovacích informácií priamo na Linux host-e. Treba paketom zabrániť traverzovanie mimo simuláciu.

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) daľ rozdeliť do niekoľkých skupín a takýmto spôsobom simulovať ich komunikáciu ak sú v ceste budovy).

V neposlednom rade by bolo treba otestovať tieto scenáre s reálnymi vozidlami, zariadeniami, na reálnom mieste so skutočnými vzdialenosťami a rýchlosťami.

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).

Ad-hoc informácie -

[2] URL: https://en.wikipedia.org/wiki/Vehicular_ad_hoc_network

[3] Elsevier, B. V.; sciencedirect URL: <https://www.sciencedirect.com/topics/computer-science/vehicular-ad-hoc-network>

Komunikácia medzi vozidlami úvod -

[4] URL: https://en.wikipedia.org/wiki/Vehicular_communication_systems

Dopravné simulátory úvod -

[5] URL: https://en.wikipedia.org/wiki/Traffic_simulation

Základné informácie o sieťových simulátoroch úvod -

[6] URL: https://en.wikipedia.org/wiki/Network_simulation

Porovnanie simulácie a emulácie -

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

SUMO a jeho aplikácie -

[8] German Aerospace Center sumo URL: https://sumo.dlr.de/docs/SUMO_at_a_Glance.html#included_applications

OSMWebWizard -

[9] German Aerospace Center sumo URL: <https://sumo.dlr.de/docs/Tutorials/OSMWebWizard.html>

TraceExporter -

[10] German Aerospace Center sumo URL: <https://sumo.dlr.de/docs/Tools/TraceExporter.html>

sumo-gui -

[11] German Aerospace Center sumo URL: <https://sumo.dlr.de/docs/sumo-gui.html>

Netedit -

[12] German Aerospace Center sumo URL: <https://sumo.dlr.de/docs/Netedit/index.html>

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

TAP device -

[14] Maxim Krasnyansky, Maksim Yevmenkin, Florian Thiel; The kernel development community URL: <https://docs.kernel.org/networking/tuntap.html>

WAF -

[15] nsnam URL: https://www.nsnam.org/docs/release/3.7/tutorial/tutorial_8.html

ns-3 intro -

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

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

ns-3 tutoriál a ukázkový kód -

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

inštalácia a požiadavky pre ns-3 -

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

NetAnim -

[20] nsnam URL: https://www.nsnam.org/wiki/NetAnim_3.108

Porovnanie sieťových simulátorov -

[21] Kirsche, Michael stackoverflow URL: <https://stackoverflow.com/questions/43207692/choosing-between-omnet-or-ns3>

[22] NS3 Simulation Projects Team URL: <https://ns3simulation.com/network-simulators-comparison/>

Ns-3 funkcionality yanswifichannel -

[23] nsnam URL: https://www.nsnam.org/doxygen/classns3_1_1_yans_wifi_channel_helper.html

WIFI channels -

[24] Albano, Samantha; minim URL: <https://www.minim.com/blog/wifi-channels-explained-what-is-wifi-channel-width#:~:text=Essentially%2C%20WiFi%20channels%20are%20smaller,the%202.4%20GHz%20frequency%20band>

[25] Khanvilkar, S., Bashir, F., Schonfeld, D. and Khokhar, A., 2004. Multimedia networks and communication. *The Electrical Engineering Handbook*, pp.431-432.

[26] Akpaída, 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.

Ns-3 funkcionality yanswifichannel -

[27] nsnam URL: https://www.nsnam.org/doxygen/classns3_1_1_log_distance_propagation_loss_model.html#details

Ns-3 funkcionality yanswifichannel -

[28] nsnam URL: https://www.nsnam.org/doxygen/classns3_1_1_buildings_propagation_loss_model.html#details

Ns-3 funkcionality yanswifiphy -

[29] nsnam URL: https://www.nsnam.org/doxygen/classns3_1_1_yans_wifi_phy_helper.html#details

Mac vrstva -

[30] The Computer Language Co Inc. URL: <https://www.pcmag.com/encyclopedia/term/mac-layer>

Ns-3 funkcionality machelper -

[31] nsnam URL: https://www.nsnam.org/doxygen/classns3_1_1_wifi_mac_helper.html#details

WIFI AP -

[32] linksys URL: <https://www.linksys.com/us/r/resource-center/what-is-a-wifi-access-point/>

WIFI STA -

[33] nstec URL: <https://www.nstec.com/network-security/network-security-what-does-sta-stand-for/>

WIFI ad-hoc -

[34] Moumita; tutorialspoint URL: [https://www.tutorialspoint.com/what-is-ad-hoc-network#:~:text=An%20ad%20hoc%20network%20is,local%20area%20networks%20\(LANs\).](https://www.tutorialspoint.com/what-is-ad-hoc-network#:~:text=An%20ad%20hoc%20network%20is,local%20area%20networks%20(LANs).)

Ns-3 funkcionality globalrouting -

[35] nsnam URL: https://www.nsnam.org/doxygen/classns3_1_1_ipv4_global_routing.html#details

Ns-3 funkcionality internetstackhelper -

[36] nsnam URL: https://www.nsnam.org/doxygen/classns3_1_1_internet_stack_helper.html#details

Ns-3 funkcionality onoffapplication -

[37] nsnam URL: https://www.nsnam.org/doxygen/classns3_1_1_on_off_application.html#details

Ns-3 funkcionality tapbridge jeho spojenie s reálnym svetom-

[38] nsnam URL: https://www.nsnam.org/doxygen/classns3_1_1_tap_bridge.html#details

Ns-3

[39] nsnam URL: https://www.nsnam.org/docs/release/3.10/doxygen/group__tap_bridge_model.html

Wireshark -

[40] Wireshark URL: <https://www.wireshark.org/>

WSL -

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

[42] URL: https://en.wikipedia.org/wiki/Windows_Subsystem_for_Linux

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

Virtual machine a WSL –

[44] DAVID DELONY; makeuseof URL: <https://www.makeuseof.com/linux-virtual-machine-or-wsl/>

Display server –

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

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

Linux kontajner –

[47] linuxcontainers Canonical Ltd. URL: <https://linuxcontainers.org/>

Ns-3 funkcionality rangepropmodel -

[48] nsnam 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.](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.)

2G3G -

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

OLSR -

[50] T. Clausen, Ed, P. Jacquet, Ed.; Network Working Group URL: <https://www.ietf.org/rfc/rfc3626.txt>

Príloha A: plán práce na projekte

1. semester

1-3 tyzden	3-5 tyzden	5-7 tyzden	7-9 tyzden	9-11 tyzden
inštalovanie potrebného softvéru	získovanie schopností SUMO a ns-3	komunikácia medzi serverom a klientom	Ad - hoc network a komplexná simulácia	Spisovanie teoretickej časti

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

Najskôr som na otestovanie SUMO použil simuláciu Račianskeho mýta. Ako druhú 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 docieľiť 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 podľa 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.

2. semester

1-2 tyzden	2-6 tyzden	6-8 tyzden	8-10 tyzden	10-12 tyzden
priprava na pokracovanie	ns-3 a reálny svet kom.	testovanie	Spisovanie vysledkov	refaktorovanie kodu

Týždeň 1 – 2

Úprava a refaktORIZOVANIE KÓDU a príprava na pokračovanie v práci

Týždeň 2 – 6

Upgrade WSL1 na WSL2. Skúšanie komunikovania z reálneho Linux host do simulácie a zo simulácie na Linux host. Pomocou Linux lxc kontajnerov a neskôr cez TAP zariadenia.

Týždeň 6 – 8

Testovanie a overovanie riešenia.

Týždeň 8-10

Spisovanie výsledkov

Týždeň 10-12

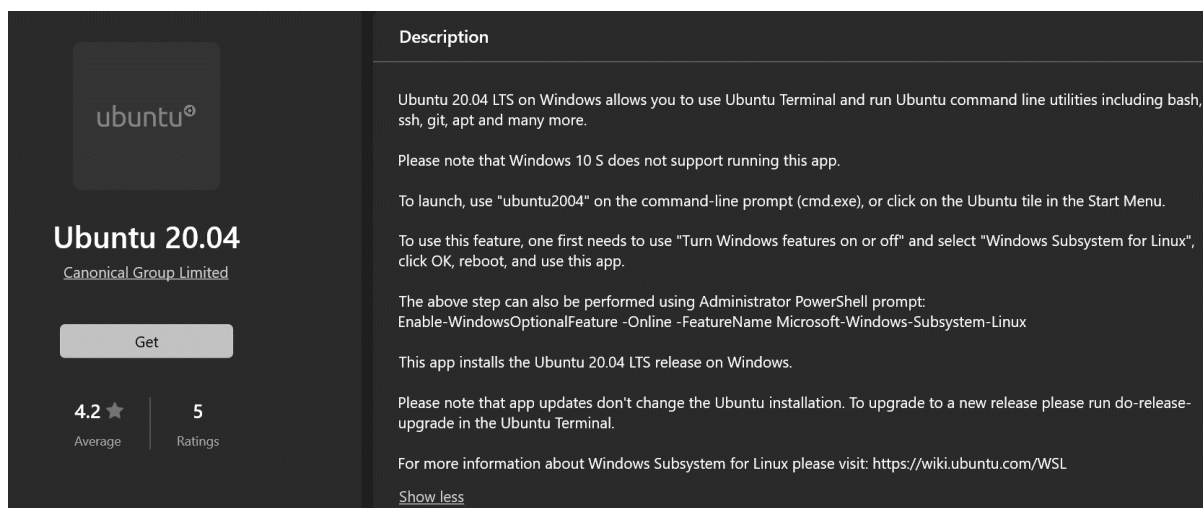
Refaktoring kódu.

Príloha B: postup inštalácie

Postup pre inštaláciu a spustenie všetkého potrebného softvéru na beh ns-3

Testovaný postup:

1. Ubuntu inštalované podľa tohto postupu z microsoft store:



Alternatívne by sa mal dať v powershell použiť príkaz:

`"wsl --install"`

tento postup sa nachádza aj tu <https://docs.microsoft.com/en-us/windows/wsl/install#set-up-your-linux-user-info>

GUI pre wsl robené podľa postupu <https://medium.com/@japheth.yates/the-complete-wsl2-gui-setup-2582828f4577> alebo z videa <https://www.youtube.com/watch?v=tmdGaXv30ug> :

Tieto príkazy sa zadávajú už v samotnom linux príkazovom riadku.

`"sudo apt update && sudo apt -y upgrade"`

`"sudo apt install build-essential"`

`"sudo apt install net-tools"`

`"sudo apt install xrdp -y && sudo systemctl enable xrdp"`

`"sudo apt install -y taskel"`

`"sudo taskel install xubuntu-desktop"` (ak sa neobjavi gui progress bar skus este raz)

`"sudo apt install gtk2-engines"`

2. Display server:

<https://sourceforge.net/projects/vcxsrv/>

Stiahnuť z tejto stránky, next, next, done.

V prípade, že bolo predtým nainštalované wsl 1, je lepšie urobiť upgrade na wsl 2 príkazom:

`"wsl --set-version Ubuntu-20.04 2"`

Možno ešte podľa microsoft návodu treba:

`"dism.exe /online /enable-feature /featurename:VirtualMachinePlatform /all /norestart"`

A stiahnuť linux kernel update package.

Návod na upgrade je tu: <https://docs.microsoft.com/en-us/windows/wsl/install-manual>

Pri každom spustení po uprade na wsl 2:

1. XLaunch – spustiť (všetko nechať zaškrtnuté a na konci zaškrtnúť disable access control)
2. Spustiť powershell (ako admin)
3. Vypni firewall
4. Zadať príkazy v powershell:

`"wsl"`

`"cat /etc/resolv.conf"` - treba skopirovať ip pre nameserver a dať ju do export display s 0 na konci teda napr:

`"export DISPLAY=172.31.16.1:0"`

`"startxfce4"`

SUMO inštalácia podľa videa <https://www.youtube.com/watch?v=tmdGaXv30ug> :

1. Príkazy v linux príkazovom riadku v domovskom adresári:

`"sudo apt update"`

`"sudo apt-get install cmake python g++ libxerces-c-dev libfox-1.6-dev libgdal-dev libproj-dev libgl2ps-dev swig"`

`"sudo apt install git"`

`"git clone --recursive https://github.com/eclipse/sumo"`

`"export SUMO_HOME="$PWD/sumo""`

`"mkdir sumo/build/cmake-build && cd sumo/build/cmake-build"`

`"cmake ../.."`

`"make -j8"` j prepínač nie je povinný a hovorí na koľkých jadrách bude bežať

NS3 inštalácia z videa https://www.youtube.com/watch?v=cPpJ_mJLkzo :

1. Spustiť wsl
2. Stiahnuť ns3 zo stránky: www.nsnam.org

3. Zadať tieto príkazy v linux príkazovom riadku:

`"sudo apt update"`

`"sudo apt install build-essential autoconf automake libxmu-dev python3-pygraphviz cvs mercurial bzip2 git cmake p7zip-full python3-matplotlib python-tk python3-dev qt5-qmake qt5-default gnuplot-x11 wireshark"`

4. Z downloads adresáru vo wsl zobrať stiahnuté ns3-allinone a umiestniť napríklad do domovského adresáru.
5. Extract here pre ns3-allinone súbor
6. V ns3-allinone adresári príkaz:

`"./build.py --enable-examples --enable-tests"`

NS3 + SUMO postup:

1. Spustiť sumo-gui
2. Pomocou select area a generate scenario sa vygeneruje potrebná časť mapy s vozidlami
3. V adresári `"~/sumo/tools/"` označené timestampom napr. `"2021-11-23-10-50-30"` by sa mal nachádzať vygenerovaný scenár.
4. Príkaz, ktorý vygeneruje xml súbor v adresári `"~/sumo/tools/2021-11-23-10-50-30"`:
`"sumo -c osm.sumocfg --fcd-output meno_suboru.xml"`
5. Príkaz, ktorý vygeneruje tcl súbor v adresári `"~/sumo/tools/"`:
`"python traceExporter.py -i 2021-11-23-10-50-30/ meno_suboru.xml --ns2mobility-output=2021-11-23-10-50-30/meno_suboru_2.tcl"`
6. Takto vytvorený tcl súbor sa dá použiť v C++ kóde s ns3 knižnicami.
7. Zdrojový kód by sa mal vytvoriť v adresári `"/ns-allinone-3.34/ns-3.34/scratch/"`
8. Na spustenie potom stačí ísť do adresáru `"/ns-allinone-3.34/ns-3.34/"` a zbehnúť príkaz:
`"./waf --run scratch/meno.cc"`
9. V prípade debuggovania sa v adresári `"/ns-allinone-3.34/ns-3.34/"` použije príkaz:
`"./waf --run scratch/meno --command-template="gdb %s """, potom run a poprípade backtrace`

Príloha C: Popis kódu

main() – inicializácia, načítanie tcl súboru, prepojenie jednotlivých nezávislých častí simulácie.

Vo funkcii main sa najprv spracujú všetky vstupné parametre. Potom sa načíta súbor obsahujúci rozmietnenia a pohyb uzlov v simulácii:

```
CommandLine cmd (_FILE_);
cmd.AddValue ("mode", "Mode setting of TapBridge", mode);
cmd.AddValue ("tapName", "Name of the OS tap device", tapName);
cmd.AddValue ("udpgen", "Generates outbound UDP traffic", udpgen);
cmd.AddValue ("moving", "Generates moving nodes", moving);
cmd.AddValue ("olsrprot", "OLSR protocol is used", olsrprot);
cmd.AddValue ("multiple_aps", "Multiple APs for multiple STAs", multiple_aps);
cmd.AddValue ("int_to_int", "Interfaces will be able to communicate", int_to_int);
cmd.Parse (argc, argv);

GlobalValue::Bind ("SimulatorImplementationType", StringValue ("ns3::RealtimeSimulatorImpl")); //to make sure that the simulation is real time
GlobalValue::Bind ("ChecksumEnabled", BooleanValue (true)); //to make sure that the simulation is real time
Config::SetDefault ("ns3::ArpCache::DeadTimeout", TimeValue (Seconds (1))); //resending ARP packets !!! otherwise the default is 1000s which is too much and the resend never happens

if (moving == "yes"){
    mobility_file = "/home/jakub/sumo/tools/2021-11-01-11-08-58/final_3_aps.tcl";
    std::cout << "moving vehicles" << std::endl;
}
else{
    mobility_file = "/home/jakub/sumo/tools/2021-11-01-11-08-58/final_3_aps_simple.tcl";
    std::cout << "stationary vehicles" << std::endl;
}
```

Potom nasleduje vytváranie skupín zariadení, ktoré by mali byť schopné spolu komunikovať:

```
for (int i = 0; i < num_of_nets; i++){
    if (i == 0){
        net_obj[i].set_options(olsrprot, mode, tapName, multiple_aps, "10.1.1.0", "10.1.3.0", "10.1.2.0");
    }
    else{
        net_obj[i].set_options(olsrprot, mode, tapNamed, multiple_aps, "10.1.4.0", "10.1.6.0", "10.1.5.0");
    }
    wifiPhy = net_obj[i].highway_points_setup();
    net_obj[i].tap_setup();
    net_obj[i].cars_setup();
    net_obj[i].car_highway_comm_setup();
}
```

Každý objekt Com_group s menom net_obj predstavuje jednu takúto úplne nezávislú skupinu. Potom main obsahuje vytvorenie podsiete v prípade, že by používateľ chcel tieto dve nezávislé časti (objekty prepojiť) napríklad pri testovaní TAP do TAP komunikácie. Potom nasleduje vytvorenie aplikácie, ktorá bude z určitého uzlu posilať UDP pakety na iný uzol:

```

if (udpgen == "yes"){
    std::cout << "UDP generator on" << std::endl;
    Ptr<Node> appSource = NodeList::GetNode (13);
    OnOffHelper onoff ("ns3::UdpSocketFactory", Address (InetSocketAddress ("10.1.4.1", 9)));
    ApplicationContainer apps = onoff.Install (appSource);
    apps.Start (Seconds (3));
    apps.Stop (Seconds (100.)); //the length of simulation
}

```

Nakoniec sa ešte spustí simulácia, prebehne a skončí.

Com_group

get_car() - vráti uzly, kvôli možnosti neskoršieho spojenia pre TAP do TAP komunikácie.

highway_points_setup() – Vytvorenie vysieláčov pri diaľnici AP/STA, nastavenie kanálov ich komunikácie:

```

YansWifiChannelHelper wifiChannel; //channel which connects phy objects
wifiChannel.SetPropagationDelay ("ns3::ConstantSpeedPropagationDelayModel"); //calculates a propagation delay
wifiChannel.AddPropagationLoss ("ns3::RangePropagationLossModel" , "MaxRange", DoubleValue (4000.0)); //models the pr
wifiPhy.SetChannel (wifiChannel.Create ());

Ssid ssid = Ssid ("left");

```

tap_setup() – Na existujúcich uzloch sa vytvorí ghost node a jeho párový uzol:

```

TapBridgeHelper tapBridge (interfacesLeft.GetAddress (1)); //creates a tap bridge and specifies a gateway for the bridge
tapBridge.SetAttribute ("Mode", StringValue (mode)); //uses a mode which creates the tap device on a real host for us and destroys it after the simulation ends
tapBridge.SetAttribute ("DeviceName", StringValue (tapName));
tapBridge.Install (nodesLeft.Get (0), devices1.Get (0)); //installs a tap bridge on the specified node and forms the bridge with the specified net device
Ptr<Node> appSink = nodesLeft.Get (0);
Ipv4Address remoteAddr = appSink->GetObject<Ipv4> ()->GetAddress (1, 0).GetLocal ();
std::cout << remoteAddr << std::endl;
std::cout << "gateway and ghost node addr"<< std::endl;
return;

```

cars_setup() – Vytvorenie uzlov, ktoré budú mobilné a budú predstavovať vozidlá. Budú komunikovať cez ad-hoc, nastavujú sa tu aj vlastnosti komunikačného kanálu, IP...

```

nodesCars.Create (2);
car_bridge = nodesCars;
WifiHelper wifi2;

wifiMac.SetType ("ns3::AdhocWifiMac");
wifi2.SetRemoteStationManager ("ns3::ConstantRateWifiManager",
    | | | | | | | | | | "DataMode", StringValue ("OfdmRate54Mbps"));

wifiPhy.SetPcapDataLinkType (WifiPhyHelper::DLT_IEEE802_11_RADIO);
YansWifiChannelHelper wifiChannel2;
wifiChannel2.SetPropagationDelay ("ns3::ConstantSpeedPropagationDelayModel");
wifiChannel2.AddPropagationLoss ("ns3::LogDistancePropagationLossModel" , "ReferenceDistance", DoubleValue (10.0));
wifiPhy.SetChannel (wifiChannel2.Create ());
NetDeviceContainer backboneDevices2 = wifi.Install (wifiPhy, wifiMac, nodesCars);

InternetStackHelper internet;
if (olsrprot == "yes"){
    | | | | | internet.SetRoutingHelper (olsr); // has effect on the next Install ()
}
internet.Install (nodesCars);

ipv4.SetBase (add_cars, "255.255.255.0");
ipv4.Assign (backboneDevices2);
return;

```

car_highway_comm_setup() – Tu sa spoja do podsiete jedno vozidlo a jeden vysielac alebo všetky vozidlá STA a vysielace AP, tiež sa tu nastavujú vlastnosti kom. kanálu a IP adresy:

```

YansWifiChannelHelper wifiChannel3;
wifiChannel3.SetPropagationDelay ("ns3::ConstantSpeedPropagationDelayModel");
wifiChannel3.AddPropagationLoss ("ns3::LogDistancePropagationLossModel" , "ReferenceDistance", DoubleValue (10.0));
wifiPhy.SetChannel (wifiChannel3.Create ());

Ssid ssid3 = Ssid ("center");
WifiHelper wifi3;

wifi3.SetRemoteStationManager ("ns3::ArfWifiManager");

ipv4.SetBase (add_comb, "255.255.255.0");

if (multiple_aps == "yes"){
    wifiMac.SetType ("ns3::ApWifiMac",
        | | | | | "Ssid", SsidValue (ssid3));
    devices2 = wifi3.Install (wifiPhy, wifiMac, NodeContainer (nodesSTA.Get(1), nodesSTA.Get(2), nodesSTA.Get(3)));

    wifiMac.SetType ("ns3::StaWifiMac",
        | | | | | "Ssid", SsidValue (ssid3),
        | | | | | "ActiveProbing", BooleanValue (false));
    devices2.Add (wifi3.Install (wifiPhy, wifiMac, nodesCars));
    Ipv4InterfaceContainer interfacesLeft3 = ipv4.Assign (devices2);
    std::cout << "multiple APs" << std::endl;
}
else {
    wifiMac.SetType ("ns3::ApWifiMac",
        | | | | | "Ssid", SsidValue (ssid3));
    devices2 = wifi3.Install (wifiPhy, wifiMac, nodesSTA.Get(2));

    wifiMac.SetType ("ns3::StaWifiMac",
        | | | | | "Ssid", SsidValue (ssid3),
        | | | | | "ActiveProbing", BooleanValue (false));
    devices2.Add (wifi3.Install (wifiPhy, wifiMac, nodesCars.Get(1)));
    Ipv4InterfaceContainer interfacesLeft3 = ipv4.Assign (devices2);
    std::cout << "single AP" << std::endl;
}

```

Spustenie simulácie:

tapname – zmení meno TAP zariadenia 1

tapname2 – zmení meno TAP zariadenia 2

udpgen – simulácia s generovaním UDP paketov

moving – simulácia s pohyblivými uzlami (autami)

olsrprot – simulácia s použitím OLSR routovacieho protokolu

multiple_aps – simulácia kde každý vysielateľ vie komunikovať s každým autom

int_to_int – simulácia kde sú dve časti prepojené a TAP 1 by môže komunikovať s TAP 2

osm.sumocfg -

```
<input>
  <net-file value="osm.net.xml"/>
  <route-files value="osm.passenger.trips_mult_car.xml"/>
  <additional-files value="osm.poly.xml"/>
</input>
```

Tu sa zadávajú súbory z ktorých sa potom generuje samotná SUMO simulácia.

osm.net.xml –

Tu sa dajú nastaviť vlastnosti jednotlivých typov ciest , povolené vozidlá na ceste podľa typu a iné:

```
<type id="highway.bridleway" priority="1" numLanes="1" speed="2.78" allow="pedestrian" oneway="1" width="2.00"/>
<type id="highway.bus guideway" priority="1" numLanes="1" speed="27.78" allow="bus" oneway="1"/>
<type id="highway.cycleway" priority="1" numLanes="1" speed="8.33" allow="bicycle" oneway="0" width="1.00"/>
<type id="highway.footway" priority="1" numLanes="1" speed="2.78" allow="pedestrian" oneway="1" width="2.00"/>
```

Tu sa upravujú informácie o samotných jazdných pruhoch na ceste:

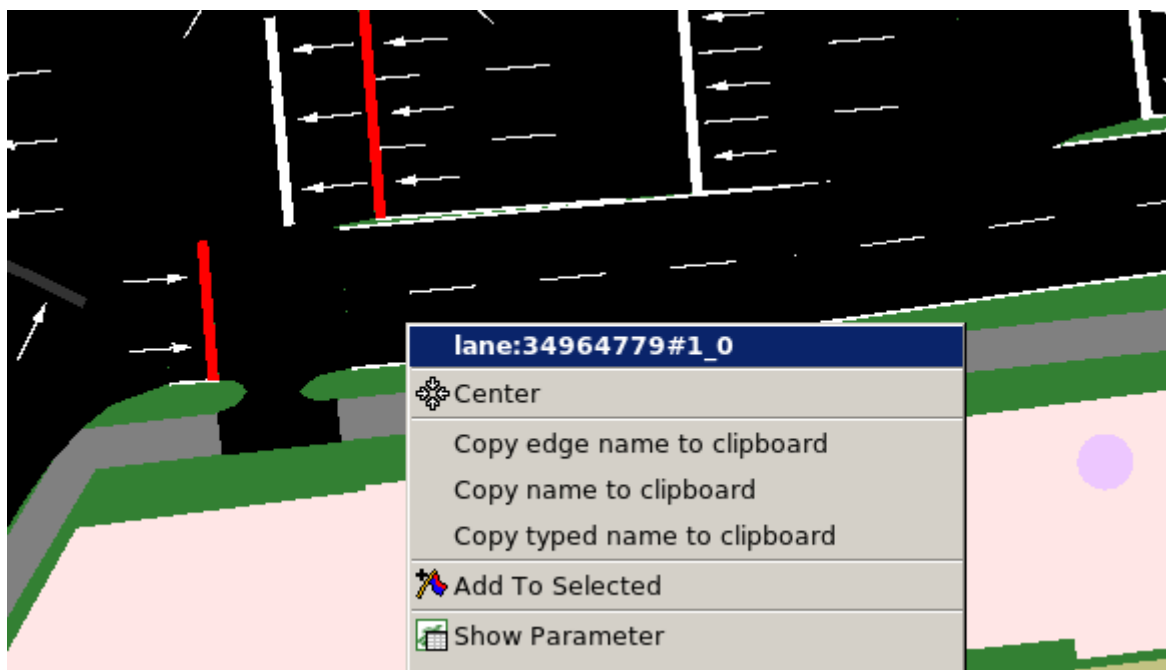
```
</edge>
<edge id="340934931#0" from="32985364" to="3481547222" priority="1" type="highway.track" shape="0.00,199.25 12.67,215.72 42.43,238.38 102.26,281.06 118.36,292.93 124.13,308.10 127.15,321.59 155.93,345.82 234.09,397.56 324.28,456.48 335.95,464.74 382.01,500.69 463.32,552.56 525.18,595.99 577.85,632.57"/>
  <lane id="340934931#0 0" index="0" allow="pedestrian motorcycle moped bicycle" speed="5.56" length="681.42" shape="1.27,198.28 13.01,214.57 43.37,237.09 103.20,279.77 119.68,291.92 125.67,307.63 128.59,320.71 156.89,344.53 234.97,396.23 325.17,455.15 336.91,463.45 382.93,499.38 464.21,551.23 526.10,594.68 539.53,604.01"/>
    <param key="origid" value="340934931"/>
  </lane>
</edge>
```

osm.passenger.trips.xml –

Tu sa nastavujú vozidlá, ich typ, jazdný pruh odkiaľ odchádzajú/prichádzajú, čas odchodu, typy vozidiel sa dajú nájsť v SUMO dokumentácii:

```
<vType id="veh_passenger" vClass="passenger"/>
<trip id="veh0" type="veh_passenger" depart="0.00" departLane="best" departSpeed="max" from="24241797#0" to="-168942727#1"/>
<trip id="veh1" type="veh_passenger" depart="45.56" departLane="best" departSpeed="max" from="-55741054" to="-743631784"/>
<trip id="veh2" type="veh_passenger" depart="91.12" departLane="best" departSpeed="max" from="316746323#0" to="24990040"/>
<trip id="veh3" type="veh_passenger" depart="136.69" departLane="best" departSpeed="max" from="4610362#0" to="23809670#3"/>
<trip id="veh7" type="veh_passenger" depart="318.93" departLane="best" departSpeed="max" from="405101794" to="-156364570#1"/>
```

Tu sa dá manuálne generovať premávka vozidiel. ID pruhu from/to sa dá zistiť kliknutím pravého tlačítka na požadovaný pruh v sumo-gui:



Viac informácii k osm.passenger.trips.xml na:

https://sumo.dlr.de/docs/Definition_of_Vehicles%2C_Vehicle_Types%2C_and_Routes.html .

