

**Slovenská technická univerzita v Bratislave
Fakulta informatiky a informačných technológií**

Tímový projekt I

Dokumentácia k projektu

Tímový e-mail

tp.team8.22@gmail.com

Členovia tímu

Bc. Andrej Dubovský

xdubovsky@stuba.sk

Bc. Samuel Kačeriak

xkaceriak@stuba.sk

Bc. Ondrej Martinka

xmartinkao@stuba.sk

Bc. Vojtech Fudaly

xfudaly@stuba.sk

Bc. Richard Andrášik

xandrasik@stuba.sk

Bc. Roman Grom

xgromr@stuba.sk

Obsah

1	Náš Tím	5
1.1	Nadobudnuté skúsenosti a znalosti	5
1.2	Členovia	6
2	Metodiky	8
2.1	Metodika SCRUM	8
2.2	Metodika Git	9
3	Analýza	13
3.1	Eclipse SUMO	13
3.2	SDN	19
3.3	NS-3	20
3.4	VANET	29
4	Návrh aplikácie	34
4.1	Logický model aplikácie	34
4.2	Drôtené modely	34
5	Prototyp aplikácie	38
5.1	Vybrané technológie	38
5.2	Front-end	38
5.3	Kontajner a jeho atribúty	39
5.4	Aplikácie	40
5.5	Back-end	42
5.6	Príručka užívateľa	43
6	Šprinty	47
6.1	Šprint 1 – Operation Homemasters	47
6.2	Šprint 2 – Operation Bonfire	49
6.3	Šprint 3 – Operation Men At Work	51
6.4	Šprint 4 – Corporate Espionage	53
7	Zápisnice	56
7.1	Zápisnica – stretnutie 1	56
7.2	Zápisnica – stretnutie 2	56
7.3	Zápisnica – stretnutie 3	57

7.4 Zápisnica – stretnutie 4	59
7.5 Zápisnica – stretnutie 5	59
7.6 Zápisnica – stretnutie 6	60
7.7 Zápisnica – stretnutie 7	61
7.8 Zápisnica – stretnutie 8	62
7.9 Zápisnica – stretnutie 9	63
7.10 Zápisnica – stretnutie 10	64

1 NÁŠ TÍM

Všetci členovia tímu sa poznáme už veľmi dlho. Na prvých tímových projektoch sme pracovali už počas strednej školy. Následne sme zostali súdržní aj po príchode na vysokú školu, kde sme v menších skupinkách pracovali na spoločných zadaniach. Spoluprácu na tímových projektoch máme preto rokmi odskúšanú a navzájom vieme, čo očakávať. Taktiež vieme zaručiť, že tento tím zostane súdržný počas celej doby trvania projektu.

Každý z nás má unikátne kvality a schopnosti, ktorými sa dokážeme navzájom dopĺňať a využiť ich v prospech tímu. Rovnako máme overené aj krízové situácie, kde sme otestovali súdržnosť tímu pod stresom. Je skvelé, že ak niekto v skupine potrebuje pomoc, ostatní členovia neváhajú a ochotne pomôžu. Ako skupina sa pravidelne stretávame aj mimo školy a máme veľa spoločných mimoškolských aktivít, ktoré ešte viacej utužujú náš kolektív. Už viackrát sme sa s kolegami rozprávali, že keby sme sa navzájom nepodporovali, tak už dávno by sme na tejto škole neštudovali.

1.1 Nadobudnuté skúsenosti a znalosti

Počas bakalárskeho štúdia členovia nášho tímu absolvovali viacero predmetov z rozličných oblastí IT priemyslu, vďaka ktorým sme nadobudli veľmi široké spektrum vedomostí o týchto témach. Medzi nimi napríklad:

- Dátová analýza a strojové učenie na predmetoch IAU a UI
- Webové aplikácie s použitím moderných js rámcov ako quasar , vue.js alebo adonis na predmete VPWA
- Mobilné aplikácie na predmete MTAA
- Blockchain developmentom pomocou web3.js alebo hyperledger SDK na predmete DMBLOCK
- Sieťové technológie na predmetoch PSIP a WANT
- Počítačová grafika na predmete PPGSO
- Znalosti operačných systémov na predmete OS
- Databázové systémy na predmete DBS

Takto sme získali veľmi cenné vedomosti v programovacích jazykoch C, C++, C#, Python(django, scapy), SQL, Java, JS(web3, vue), TS(quasar, adonis), Php(laravel), Kotlin, solidity a vo vývoji aplikácii v Unix aj Windows prostredí. Nebojíme sa tento zoznam obohatiť aj o nové jazyky a technológie. Táto rozmanitosť záujmov pokračuje aj pri štúdiu inžinierskeho stupňa. Tu sa jeden člen tímu rozhodol prehlbovať svoje vedomosti v oblasti informačnej bezpečnosti a má zapísane predmety ako penetračné testovanie alebo bezpečnosť v informačných technológiách. Zvyšok tímu pokračuje v zdokonaľovaní sa v oblasti softvérového inžinierstva. Zamieriava sa na najmä na moderné postupy umelej inteligencie a dátovej analýzy. Pričom majú premety Neurónové siete, Počítačové videnie alebo Vyhľadávanie informácií.

1.2 Členovia

Ondrej:

Nenechajte sa oklamať jeho na prvý pohľad tichším vonkajším pôsobením. V oblasti tímového projektu pôsobí ako neoblomný diktátor, ktorému nevedno protirečiť. Každú chybu trestá nemilosrdne. Riadi sa životným mottom: „Škola je celý môj život“. Preto celé jeho štúdium smeruje k jedinému cieľu. A to dosiahnutiu doktorandského stupňa štúdia.

Roman:

Len málokto vie o športových udalostiach a pive viac ako on. Dostal požehnanie múdrosti v mnohých oblastiach života. Tieto vedomosti potom plne prezentuje pri kariére profesionálneho riešiteľa kvízov. Škoda len že nevie ako vyzerá Jozef Miloslav Hurban. Už pred mnohými rokmi objavil medzeru na trhu a tou sa aktuálne aj živý. Stávkovanie, automaty a odhrňanie snehu.

Samuel:

Mimo jeho dosahu je nazývaní aj „Pastier“. Túto prezývku si získal pre jeho schopnosť ovplyvňovať životy. Preto kým nie ste si istý na 100% vašim názorom, tak sa k nemu radšej ani nepribližujte. Miluje drámu. Teda pokiaľ sa ho netýka. Rád investuje do bicyklov ale neznáša drahé jedlá. Jeho super schopnosťou je, že vie nájsť zlú vlastnosť na hocičom a hocikom.

Vojtech:

Áno je to on. Mnohé sa nás pýtajú, či sú všetky chýry, ktoré o ňom kolujú pravdivé. S istotou môžeme len potvrdiť, že nikto okrem neho to nevie naisto. Ak máte to šťastie a máte frajerku, odporúčame aby ste ju držali od neho čo najďalej, pokiaľ o ňu nechcete prísť. Patrí medzi „intrákovú“ elitu a rád chodí do kotolne. Aj keď nikto iný s tímu nevie, čo presne tam robí.

Richard:

V projekte bol jednohlasne zvolený, kvôli svojej nadmernej dominancii, za Scrum mastera. Avšak v realite môžeme povedať, že ak je on hlava, tak Ondro je krk ktorý s ním krúti a Samo nohy, ktoré ním hýbu. Je psychicky naviazaný na svojho „Jaštera“, s ktorým rád trávi čas osamote. Miluje bubnovanie a trochu sa ponáša na piráta, aj so všetkým, čo k tomu patrí.

Andrej:

No a je to tu zas. Vďaka jeho očarujúcej bakalárskej práci bol požiadaní, aby pre fakultu natočil reklamné video na Instagram. Od vtedy má nos vystrelený niekde v oblakoch. Miluje akýkoľvek šport a vždy sa tvári ako najlepší. Aj keď všetci vedia že to tak nie je. Jeho silnou stránkou sú jeho skúsenosti a znalosti v oblasti vzťahov. Tak ak máte nejaký takýto problém, kľudne sa na neho obráťte.

2 METODIKY

2.1 Metodika SCRUM

Náš tímový projekt sme vyvíjali pomocou metodiky SCRUM. V tejto metodike definujeme pravidlá a opíšeme techniky pri používaní SCRUM. Za správne dodržiavanie metodiky SCRUM je zodpovedný Scrum Master.

Agile

Projekt vyvíjame metódou Agile. Pri Agile si plánujeme úlohy dynamicky a podľa priorít. Plánovanie musí byť vždy adaptívne na aktuálnu situáciu aby pokiaľ sa niečo zmení dokázali by sme sa flexibilne prispôbiť novej situácii. Na rozdiel od waterfall techniky, úlohám sú rozdelené priority pomocou plánovacích kartičiek a pri prideľovaní úloh členom tímu má posledné slovo Scrum Master tímu.

Scrum Master

Hlavnou úlohou Scrum Mastera je dodržiavanie Agile a Scrum metodík a kontrolovanie a monitorovanie sprintov. Sprint je časové obdobie na ktoré sa rozdelia úlohy, ktoré v rámci tohto obdobia musia byť vypracované. Každý sprint trvá dva týždne. V našom tíme Scrum Master vypracováva po dokončení každého sprintu aj reflexiu sprintu.

Nástroj Azure DevOps

Na monitorovanie úloh a Sprintov aj na sledovanie všetkých ostávajúcich úloh používame nástroj Azure DevOps. Do DevOps je možné vytvoriť “backlog” úloh, teda zoznam úloh, ktoré je potrebné ešte splniť. Pri začiatku každého sprintu povyberáme vhodné úlohy z backlogu, rozdelíme ich na menšie “tasky” a pridáme tasky členom tímu pomocou DevOps. Každý task rieši práve jeden člen tímu. Po dokončení tasku ho môžeme označiť ako dokončený. Ideálne chceme dosiahnuť aby sme za sprint dokončili všetky pridelené tasky.

Organizácia sprintu

Každý sprint trvá 2 týždne a sú počas neho dve osobné stretnutia všetkých členov tímu na FIIT. V prvom stretnutí si zvyčajne vyberieme a rozdelíme úlohy a určíme cieľ sprintu. V druhom stretnutí, ktoré je v strede sprintu si hovoríme, na čom sme pracovali a rozdelíme úlohám nové priority. Po sprinte je revízia sprintu, ktorá sa koná v ten istý deň ako prvé stretnutie nasledujúceho sprintu.

Počas sprintu je každý člen tímu povinný prideľovať si tasky na DevOps a definovať im adekvátne výstupy. Po vypracovaní výstupu sú členovia tímu povinný presunúť ten task do kolónky hotových taskov, aby sa nám to odzrkadlilo na burndown grafe. Cez sprint sú úlohy Scrum Mastera: vytvoriť tasky a dohliadnuť aby boli správne pridelené, kontrolovať plnenie taskov a aby mali adekvátny výstup, uzatvorenie sprintu po ukončení.

Retrospektíva sprintu

Počas sprintu sa spisuje dokument o retrospektíve sprintu. Tento dokument vždy píše Scrum Master. Skladá sa z piatich častí: tabuľka o prehľade sprintu, začiatok sprintu, rekapitulácia v strede sprintu, reflexia sprintu a finálny burndown graf z Azure DevOps.

Tabuľka obsahuje jednoducho spísané informácie o tom, kto čo robí v prvom týždni sprintu a v druhom týždni sprintu. Taktiež je tam percentuálne vyjadrenie, kto koľko svojej práce urobil každý týždeň. V začiatku sprintu je napísané, čo každým sprintom chceme dosiahnuť, kto bude približne akú prácu robiť a sú tam aj informácie k organizácii sprintu. Rekapitulácia v strede sprintu sa robí počas stretnutia presne v strede každého sprintu. Je tam spomenuté, kto akú prácu urobil a s čím bude pokračovať ďalší týždeň.

Reflexia sprintu obsahuje podrobne spísané každým z nás dosiahnuté ciele. Sú tam aj spomenuté zistenia k organizácii sprintov a či sme dosiahli, všetko, čo sme počas sprintu chceli. Burndown graf je z nástroja Azure DevOps. Je na ňom možné vidieť proces robenia úloh počas celého sprintu.

2.2 Metodika Git

Github je online nástroj na verzionovanie projektov. Okrem samotného verzionovania ponúka množstvo iných funkcionalít ako napríklad bug tracking, task management a ďalšie. Na tomto projekte sme sa rozhodli používať tento nástroj hlavne kvôli týmto funkciám a dobrých predchádzajúcim skúsenostiam. Projekt máme rozdelený do troch repozitárov: web, aplikačný frontend a aplikačný backend. V každom repozitári sa nachádza súbor .gitignor slúžiaci na ignorovanie súborov, ktoré nepotrebujeme uploadovať online(napríklad node_modules). Repozitáre sú verejne dostupné na odkaze: <https://github.com/vanetnuggets>. Pre efektívnejšiu spoluprácu sme si spoločne zadefinovali niekoľko pravidiel používania Githubu.

Pravidlá používania:

- do main branch sa priamo nepushuje, pridávanie nového kódu je zabezpečené pull requestami s child branchov
- každá nová funkcionality sa developuje vo vlastnej banchi
- pred mergom do main branche je potrebné dôkladné odtestovanie implementovanej funkcionality
- merge request by mal byť skontrolovaný aspoň jedným iným členom tímu
- merge vykonávame až po rebasenutí main branche aby nedochádzalo ku konfliktom

Bonusové pravidlo:

- commit musí začínať emotikonom

Uvedieme aj niekoľko príkladov použitia gitu aj s konkrétnymi príkazmi:

1. práca na feature branchi a update z main branch

tento postup využívame pri tomto projekte bežne, pretože každý člen pracuje na svojej branchi a až keď je feature hotová tak sa mergeje do main branch

- **git add .** -pridáme súbory
- **git status** -skontrolujeme či sme pridali správne súbory
- **git commit -m "<opis čo sa zmenilo>"** -commit s výstižným popisom
- **git fetch** -stiahneme si updaty z remote branch
- **git stash** -ak sme nepridávali všetky súbory, musíme urobiť stash
- **git rebase origin/main** -aplikujeme náš commit na main branch aby nedošlo ku konfliktom
- **git push -f origin <feature_branch>** -ak sme dávali rebase tak musíme použiť prepínač f, pretože sme updatli main
- **git stash apply** -ak sme si predtým stashli súbory teraz si ich vieme znova aplikovať

2. vytvorenie feature_branch

takýmto spôsobom vieme vytvoriť separátny branch na ktorom, vieme implementovať novú funkcionality

- **git checkout -b <feature_branch>**

3. prepínanie medzi branchami

nasledujúcim príkazom sa vieme prepínať medzi jednotlivými branchami, ak máme necommitnuté súbory najskôr je potrebné vykonať príkaz stash

- **git checkout <branch_name>**

4. merge request

- merge request používame po implementovaní a otestovaní novej funkcionality na našej child branchi, pre merge do main branch nastavíme aspon jedného reviewera a počkame na schvalenie, po schvalení a za predpokladu nevzniknutia konfliktov môžeme branch mergnut. Takto vieme minimalizovať riziko bugov v main branch.

3 ANALÝZA

3.1 Eclipse SUMO

„Simulácia mestskej mobility“ alebo skráteno „SUMO“ je mikroskopická, multimodálna simulácia dopravy s otvoreným zdrojom. Umožňuje simulovať, ako sa daný dopravný dopyt, ktorý pozostáva z jednotlivých vozidiel, pohybuje po danej cestnej sieti. Simulácia umožňuje riešiť veľký súbor tém riadenia dopravy. Je čisto mikroskopická: každé vozidlo je explicitne modelované, má vlastnú trasu a pohybuje sa individuálne po sieti. Simulácie sú predvolene deterministické, ale existujú rôzne možnosti na zavedenie náhodnosti.

Niektoré oblasti využitia:

- Vyhodnocovať výkon semaforov vrátane vyhodnocovania moderných algoritmov až po vyhodnocovanie týždenných časových plánov.
- Skúmanie výberu trasy vozidla, vrátane vývoja nových metód, hodnotenia ekologicky orientovaného smerovania založeného na emisiách znečisťujúcich látok a skúmania vplyvov výberu autonómnej trasy na celú sieť.
- SUMO je široko používaný komunitou V2X na poskytovanie realistických stôp vozidiel a na vyhodnocovanie aplikácií v on-line slučke so sieťovým simulátorom.
- AI tréning plánov semaforov.
- Simulácia a overenie funkcie autonómneho riadenia v spolupráci s inými simulátormi.
- Simulácia parkovacej dopravy.

Komponenty:

sumo command line simulation	sumo-gui simulation with a graphical user interface	netconvert network importer	netedit visual editor for network elements
netgenerate abstract networks generator	od2trips converter from O/D matrices to trips	duarouter routes generator based on a dynamic user assignment	jtrrouter routes generator based on turning ratios at intersections
dfrouter route generator with use of detector data	marouter macroscopic user assignment based on capacity functions	polyconvert imports geometrical shapes and convert them to be visualized using sumo-gui	activitygen compute mobility wishes based on population
emissionsMap emission map generator	emissionsDrivingCycle calculates emission values based on a given driving cycle	osmWebWizard generate a SUMO scenario with just a few clicks	And many other Tools!

Vlastnosti:

- Simulácia
- Priestorovo kontinuálny a časovo diskretný pohyb vozidla
- Rôzne typy vozidiel
- Viacprúdové ulice s radením jazdných pruhov
- Rôzne pravidlá prednosti v jazde, semaforey
- Rýchle grafické užívateľské rozhranie OpenGL
- Spravuje siete s niekoľkými 10 000 okrajmi (ulíc)
- Vysoká rýchlosť vykonávania (až 100 000 aktualizácií vozidla/s na stroji s frekvenciou 1 GHz)
- Interoperabilita s inými aplikáciami za behu
- Výstupy v celej sieti, na okrajoch, na vozidlách a na detektoroch
- Import siete
- Importuje VISUM, Vissim, Shapefiles, OSM, RoboCup, MATsim, OpenDRIVE a popisy XML
- Chýbajúce hodnoty sa určujú pomocou heuristiky
- Mikroskopické trasy – každé vozidlo má svoju
- Rôzne algoritmy dynamického priradenia používateľov
- Vysoká prenosnosť
- Vysoká interoperabilita vďaka použitiu iba XML údajov
- Open source (EPL 2.0)

Inštalácia:

✓ Windows:

- Download 64 bit installer: [sumo-win64-1.14.1.msi](#) ↗
- Download 64 bit zip: [sumo-win64-1.14.1.zip](#) ↗
- Download 32 bit installer: [sumo-win32-1.14.1.msi](#) ↗
- Download 32 bit zip: [sumo-win32-1.14.1.zip](#) ↗

✓ Linux:

- `sudo add-apt-repository ppa:sumo/stable`
- `sudo apt-get update`
- `sudo apt-get install sumo sumo-tools sumo-doc`

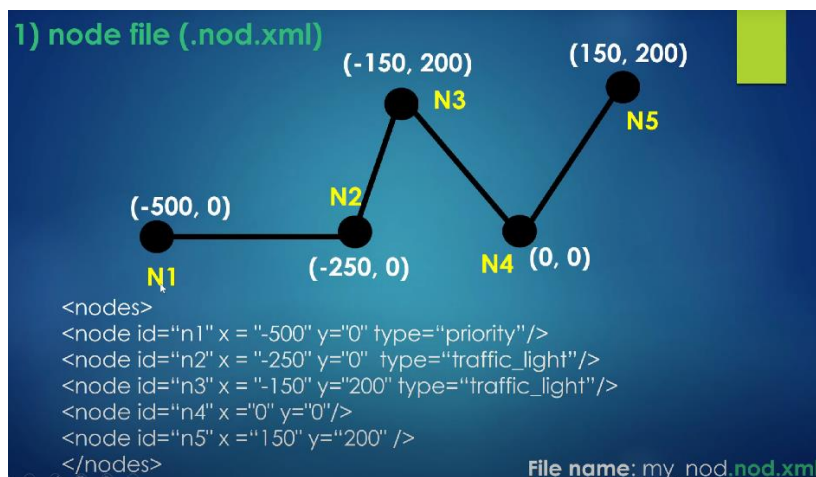
sumo:

sumo je samotná simulácia; ide o mikroskopickú, priestorovo spojitú a časovo diskretnú simuláciu dopravného toku.

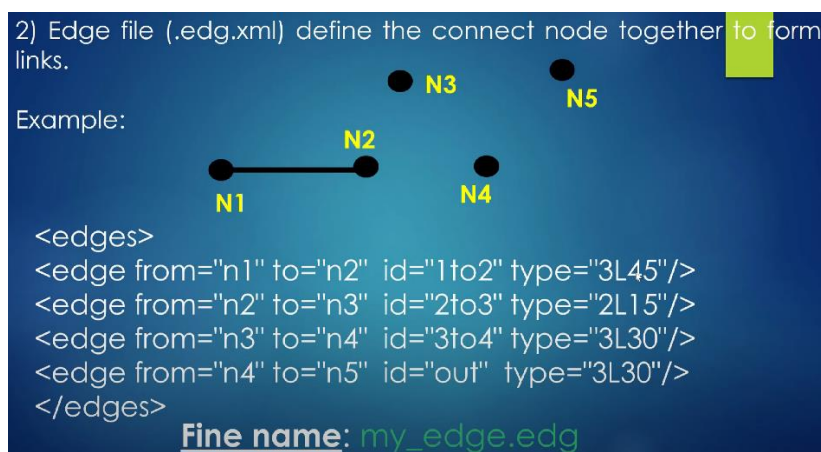
- **Účel:** Simuluje definovaný scenár

- **Systém:** prenosný (testuje sa Linux/Windows); beží na príkazovom riadku
- **Vstup (povinný):**
 - cestná sieť vytvorená prostredníctvom netconvert alebo netgenerate
 - súbor trás
- **Vstup (voliteľné):** Ďalšie definície semaforov, značiek s premenlivou rýchlosťou, výstupných detektorov atď.
- **Výstup:** SUMO umožňuje generovať širokú škálu výstupov; vizualizácia sa vykonáva pomocou sumo-gui
- **Programovací jazyk:** C++
- **Použitie:**
 - o Skladá sa zo šiestich krokov:

1.krok: vytvorenie súboru s uzlami



2.krok : vytvorenie súboru s cestami



3.krok: vytvorenie súboru s typmi ciest

3) Type file (.type.xml) include road priority, the number of lanes, speed limit, type of vehicles allow, etc.


Example:

```
<types>
<type id="3L45" priority="3" numLanes="3" speed="45"/>
<type id="2L15" priority="3" numLanes="2" speed="15"/>
<type id="3L30" priority="2" numLanes="3" speed="30"/>
</types>
```

Fine name: my_type.type

4.krok: vytvorenie siete s uzlov, ciest a ich typov

4) **netconvert**



my_nodes.nod.xml my_edge.edg.xml my_type.type.xml my_net.net.xml

```
netconvert --node-files my_nodes.nod.xml -  
-edge-files my_edge.edg.xml -t  
my_type.type.xml -o my_net.net.xml
```

5.krok: vytvorenie súboru trás

5) Route file (.rou.xml)

```
<routes>
<vType accel="1.0" decel="5.0" id="Car" length="2.0" maxSpeed="100.0" sigma="0.0" />
<vType accel="1.0" decel="5.0" id="Bus" length="12.0" maxSpeed="1.0" sigma="0.0" />

<route id="route0" edges="1to2 2to3"/>
<vehicle depart="10" id="veh0" route="route0" type="Bus" />
<route id="route1" edges="2to3 3to4"/>
<vehicle depart="10" id="veh1" route="route1" type="Car" />
<route id="route2" edges="3to4 out"/>
<vehicle depart="30" id="veh2" route="route2" type="Car" />
</routes>
```

6.krok: vytvorenie konfiguračného súbor

Lastly, the Sumo Configuration file (.sumocfg)

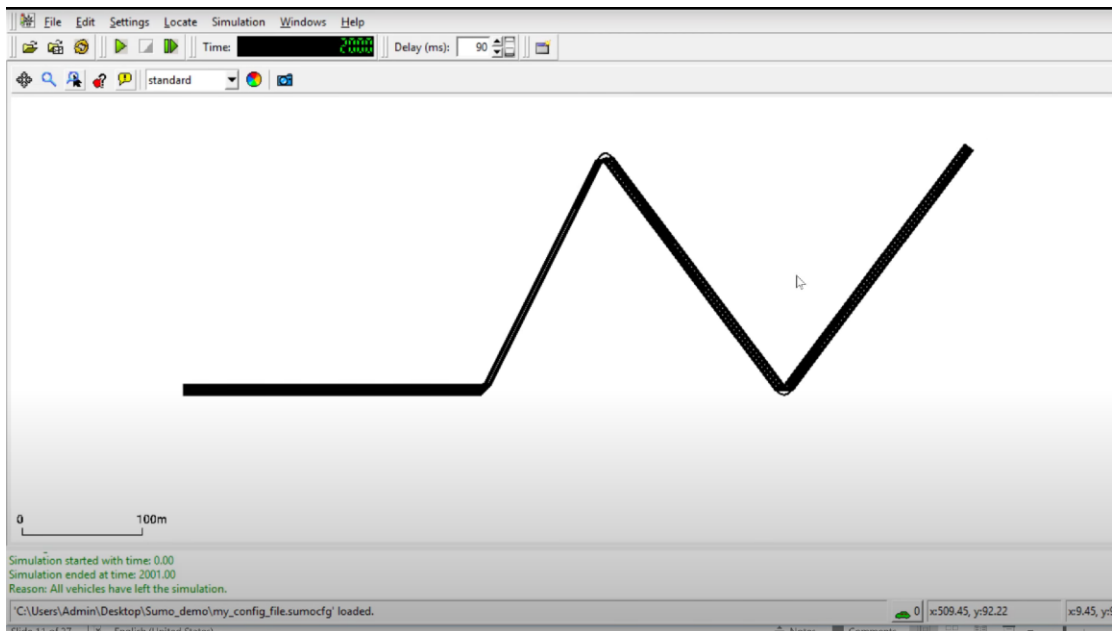
```
<configuration>

<input>
<net-file value="my_net.net.xml"/>
<route-files value="my_route.rou.xml"/>
</input>
<time>
<begin value="0"/>
<end value="2000"/>
</time>

File name: my_config_file

</configuration>
```

Po spustení konfiguračného súboru:



sumo-gui:

- sumo-gui je v podstate rovnaká aplikácia ako sumo, len je rozšírená o grafické používateľské rozhranie.
- **Účel:** Simuluje definovaný scenár
- **System:** prenosný (testuje sa Linux/Windows); otvorí okno
- **Vstup (povinný):** Konfiguračný súbor SUMO
- **Výstup:** sumo-gui generuje rovnaký výstup ako sumo
- **Programovací jazyk:** C++

netconvert:

- netconvert importuje digitálne cestné siete z rôznych zdrojov a vytvára cestné siete, ktoré môžu byť použité inými nástrojmi z balíka.
- **Účel:** Import a konverzia cestných sietí
- **System:** prenosný (testuje sa Linux/Windows); beží na príkazovom riadku
- **Vstup (povinný):** Definícia cestnej siete
- **Výstup:** Vygenerovaná cestná sieť SUMO; voliteľne aj iné výstupy
- **Programovací jazyk:** C++
- **Príkaz na vytvorenie siete s OpenStreetMap:**

netconvert --osm-files berlin.osm.xml -o berlin.net.xml

3.2 SDN

Software Defined Networking – Na rozdiel od klasického prístupu, kde sa jednotlivé routre starajú o flow paketov, v SDN má túto úlohu controller. Na jednom centralizovanom mieste vieme nakonfigurovať viacero zariadení a tým odpadá potreba konfigurácie po jednom. Ďalším rozdielom sú údaje o sieti. Zariadenia dokážu posilať údaje o aktuálnej premávke a podľa toho je možné kontrolovať samotnú sieť. Architektúra SDN sa skladá z troch vrstiev: aplikačná, kontrolerová a vrstva infraštruktúry.

V aplikačnej vrstve sa môžu nachádzať napríklad load balancery alebo firewall komunikujúce priamo s controllerom, kde pri klasickej architektúre museli byť použité špecializované zariadenia. Táto vrstva teda obsahuje už konkrétne aplikácie, ktoré majú požiadavky na sieť a pomocou druhej vrstvy komunikuje s fyzickou infraštruktúrou.

Vrstva kontroleru spočíva v centralizovanom manažovaní celej siete. Klasické na routroch beží napríklad spanning tree protocol na získanie obrazu o susedoch ale v SDN architektúre si flow paketov vieme určiť jednotlivo pomocou kontroleru, dokonca vieme aplikovať na jednotlivé typy paketov aj filtre alebo rôzne pravidlá podľa, ktorých sa budú pakety smerovať. Napríklad ak zdetekujeme VOiP packet vieme ich poslať inou cestou ako ostatné, poprípade ak zdetekujeme autentifikáciu vieme pakety presmerovať patričným smerom. Kontroler nám teda dáva väčšiu voľnosť nad správou siete a umožňuje jednoduchšiu a častejšiu konfiguráciu ak je to potrebné. Na komunikáciu medzi kontrolerom a fyzickými zariadeniami sa používa OpenFlow protokol.

Do infraštruktúrálnych vrstvy patria už samotné zariadenia, teda routre a switchy. Starajú sa o samotný flow paketov na základe pravidiel poskytnutých kontrolerom. Takisto zabezpečujú poskytovanie štatistických údajov o sieti.

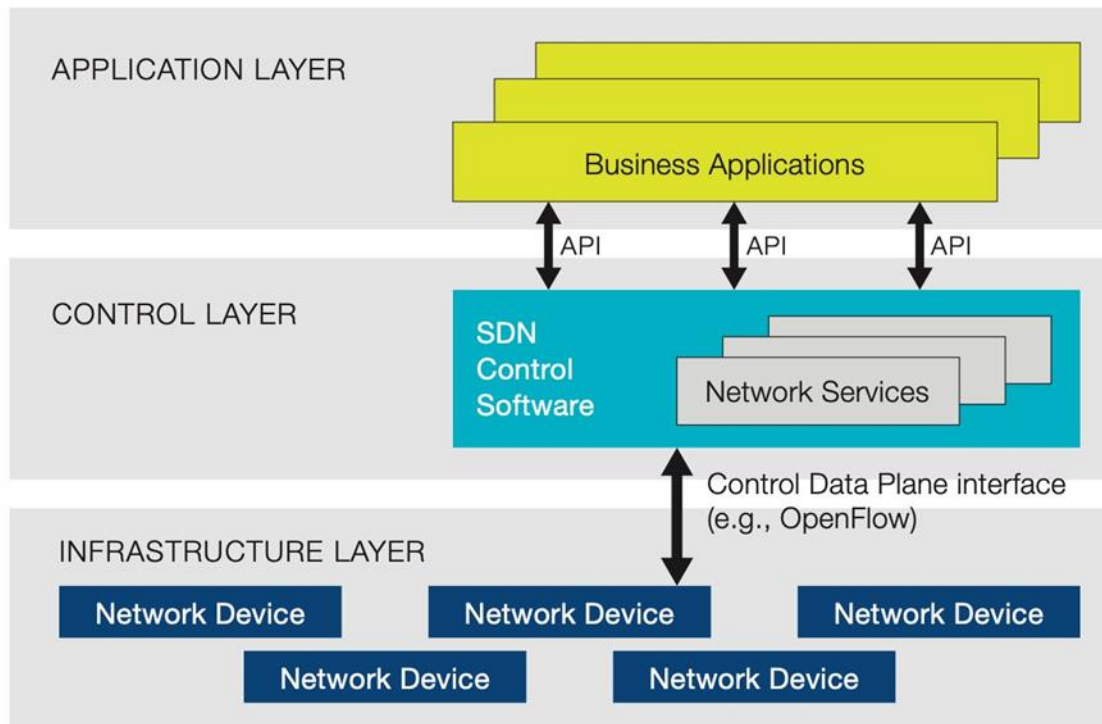
Výhody:

- Nižšia cena
- Väčšia kontrola nad sieťou
- Jednoduchšia rekonfigurácia
- Škálovateľnosť

Nevýhody:

- Rekonfigurácia celej siete

- Zmena postupov práce administrátorov -> preučenie



Zdroje

https://www.google.com/search?q=sdn+layers&tbm=isch&sxsrf=ALiCzsZ9LEtwxsBM5khalxbH9EqM6HcYA:1667313064058&source=lnms&sa=X&ved=2ahUKEwjf76WXmY37AhU_QPEDHcf8DF0Q0pQJegQIBhAE&biw=2048&bih=1010&dpr=1.25#imgsrc=h1taJGs-eV7OHM

<https://yourtechdiet.com/blogs/software-defined-networking-sdn/>

<https://www.comparitech.com/net-admin/software-defined-networking/>

3.3 NS-3

Simulátor NS-3 je spustiteľný na rôznych operačných systémoch - Windows, MacOS a linux na rôznych distribúciach s rôznymi metódami inštalácie. My v našom projekte používame linux, takže ďalšie kapitoly budú opisovať používané šimulátora na tejto platforme.

Inštalácia

Je prehľadne zdokumentovaná v ns-3 dokumentácii [1]. Použili sme manuálnu inštaláciu cez git.

1. Naklonujeme si repozitár

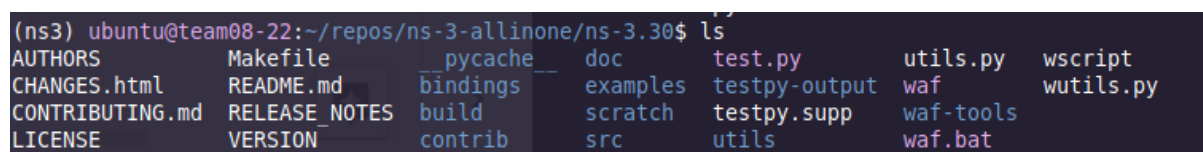
```
git clone https://gitlab.com/nsnam/ns-3-allinone.git
```

2. Po naklonovaní sa v priečinku ns-3-allinone zobrazí python script, pomocou ktorého vieme stiahnuť hocijakú verziu simulátora. My zvolíme verziu 3.30 a nainštalujeme ju príkazom `./download.py -n ns-3.30`
3. Prejdeme do adresára ns-3.30 príkazom `cd ./ns-3.30`, v ktorom už potrebujeme len zostaviť simulátor z jeho zdrojov. Je na to script `build.py`, ktorý spustíme príkazom `./build.py`

Ns-3 je následne nainštalované. Na jeho používanie ho ešte treba nakonfigurovať, čomu sa budeme venovať v nasledujúcej kapitole.

Konfigurácia NS-3

Po nainštalovaní nami zvolenej verzie ns-3 v priečinku ostane veľa rôznych adresárov a súborov, zobrazené na nasledujúcom obrázku



```
(ns3) ubuntu@team08-22:~/repos/ns-3-allinone/ns-3.30$ ls
AUTHORS      Makefile      __pycache__   doc           test.py       utils.py      wscript
CHANGES.html  README.md     bindings      examples      testpy-output  waf           wutils.py
CONTRIBUTING.md  RELEASE_NOTES  build         scratch       testpy.supp    waf-tools
LICENSE        VERSION       contrib       src           utils          waf.bat
```

Našťastie, veľa z nich nepotrebujeme vôbec riešiť. Tie podstatné si prejdeme v nasledujúcich podkapitolách

Waf je systém založený v pythone na zjednodušenie operácii s ns-3 simulátorom. Poskytuje jednoduchý systém v ktorom sa robí všetko súvisiace s ns-3 simulátorom, hlavne konfigurácia modulov ns-3, kompilácia s spúšťanie scenárov

- a) Konfigurácia modulov sa robí pomocou príkazu `./waf configure`. Tento príkaz musí byť spustený pred každou zmenou konfigurácie, inak waf bude používať tú starú. Po inštalácii je waf nenakonfigurovaný, takže to treba spustiť aspoň raz.
- b) Aby sme vedeli, či simulátor správne funguje musíme nechať zbehnúť testy. Na to najprv treba nakonfigurovať waf jednoduchým príkazom `./waf configure --enable-tests`. Následne môžeme spustiť testy príkazom `./test.py`. Testy sa začnú postupne vykonávať, ale keďže ich

je veľa tak to chvíľu trvá.

```
PASS: TestSuite routing-aodv-loopback
PASS: TestSuite routing-aodv-regression
PASS: TestSuite lte-frequency-reuse
256 of 259 tests passed (256 passed, 3 skipped, 0 failed, 0 crashed, 0 valgrind errors)
List of SKIPPed tests:
  ns3-tcp-cwnd (requires NSC)
  ns3-tcp-interoperability (requires NSC)
  nsc-tcp-loss (requires NSC)
```

Po zbehnutí testov je ns-3 pripravené na vytváranie a spúšťanie scenárov.

Spúšťanie NS-3 simulácii

Spúšťanie simulácii je jednoduché. Treba skopírovať zdrojový kód simulácie do podpriechinky ./scratch. Následne sa simulácia spustí cez waf príkazom ./waf --run <filename>. V takomto príkaze sa zdrojový kód simulácie hľadá v priečinku ./scratch. Názov súboru sa nepíše s koncovkou .cc, inak ho waf nenájde.

```
(ns3) ubuntu@team08-22:~/repos/ns-3-allinone/ns-3.30$ ./waf --run scratch/third
Waf: Entering directory `/home/ubuntu/repos/ns-3-allinone/ns-3.30/build'
[1947/1999] Compiling scratch/third.cc
[1959/1999] Linking build/scratch/third
Waf: Leaving directory `/home/ubuntu/repos/ns-3-allinone/ns-3.30/build'
Build commands will be stored in build/compile_commands.json
'build' finished successfully (5.150s)
At time 2s client sent 1024 bytes to 10.1.2.4 port 9
At time 2.01796s server received 1024 bytes from 10.1.3.3 port 49153
At time 2.01796s server sent 1024 bytes to 10.1.3.3 port 49153
At time 2.03364s client received 1024 bytes from 10.1.2.4 port 9
```

Výstup simulácie je priamo definovaný v simulačnom scenári. Medzi rôzne výstupy patria napr. .xls súbory, .pcap súbory alebo logy v textovom formáte. Tento výstup hlbšie preberieme v kapitole o simulácii.

NS-3 a python

Väčšina simulátorov je napísaná v programe c++. NS-3 dnes ponúka aj alternatívu, a to je písanie scenárov v jazyku python. Takéto scenáre majú rýchlejšie prvé spustenie, nakoľko sa nemusia kompilovať a sú jednoduchšie na písanie, lebo python je jednoduchší jazyk. Na druhú stranu k nim je menej dokumentácie.

Na spustenie ns-3 simulácie v pythone si treba najprv pripraviť prostredie. Od verzie pythonu 3.9 je jedna používaná funkcia už zastaralá a vyhadzuje varovanie. Bohužiaľ waf pri konfigurácii varovania považuje za error a preto to spoľahlivo funguje len s pythonom verzie <=3.9.

```
[1853/1997] Compiling build/src/network/bindings/ns3module.cc
src/core/bindings/ns3module.cc: In function 'PyObject* PyInit_core()':
src/core/bindings/ns3module.cc:51659:23: error: 'void PyEval_InitThreads()' is deprecated [-Werror=deprecated-declarations]
51659 |     PyEval_InitThreads();
      |     ~~~~~^
In file included from /usr/include/python3.10/Python.h:130,
               from src/core/bindings/ns3module.h:3,
               from src/core/bindings/ns3module.cc:1:
/usr/include/python3.10/ceval.h:122:37: note: declared here
 122 | Py_DEPRECATED(3.9) PyAPI_FUNC(void) PyEval_InitThreads(void);
      |                               ^~~~~~
cc1plus: all warnings being treated as errors
```

Zvolili sme teda verziu 3.8. Pridáme si do apt repozitárov repozitár deadsnakes príkazom

```
sudo add-apt-repository ppa:deadsnakes/ppa
```

Tento repozitár obsahuje staršie verzie pythonu. Ďalej nainštalujeme našu verziu cez príkaz (aj dev je potrebná, inak to nezbehne)

```
sudo apt install python3.8 python3.8-dev
```

Do pythonu potrebujeme nainštalovať venv modul na spracovanie virtuálnych prostredí príkazom

```
sudo apt install python3.8-venv
```

Následne vytvoríme virtuálne prostredie príkazom `python3 -m pip venv ns3` a aktivujeme ho príkazom

```
source <path/na/venv>/bin/activate
```

NS3 pre spúšťanie potrebuje balíček pybindgen, ktorý nainštalujeme cez pip príkazom

```
pip install pybindgen
```

Ako posledná vec je potrebné nakonfigurovať waf, aby vedela spustiť scenáre definované v pythone príkazom `python`

```
./waf configure.
```

Tento proces prvýkrát trvá dlhšie, lebo kompiluje potrebné moduly.

Prostredie je teraz nakonfigurované a môžeme spustiť scenár príkazom

```
./waf -pyrun ./path/na/scenar.py.
```



```
(ns3) ubuntu@team08-22:~/repos/ns-3-allinone/ns-3.30$ ./waf --pyrun ./examples/tutorial/third.py
Waf: Entering directory `/home/ubuntu/repos/ns-3-allinone/ns-3.30/build'
Waf: Leaving directory `/home/ubuntu/repos/ns-3-allinone/ns-3.30/build'
Build commands will be stored in build/compile_commands.json
'build' finished successfully (0.844s)
At time 2s client sent 1024 bytes to 10.1.2.4 port 9
At time 2.01796s server received 1024 bytes from 10.1.3.3 port 49153
At time 2.01796s server sent 1024 bytes to 10.1.3.3 port 49153
At time 2.03364s client received 1024 bytes from 10.1.2.4 port 9
```

Simulácia

Základom simulácie sú uzly (node). Je to abstrakcia predstavujúca nejaký počítač/stroj/zariadenie. Tieto uzly sa následne špecifikujú. V scenároch sa často stáva, že viacero uzlov môže mať rovnakú konfiguráciu. Preto vieme viacero uzlov vložiť do jedného kontajneru a “nainštalovať” im konfiguráciu naraz. Slúži na to prvok NodeContainer. Takáto konfigurácia vyzerá nasledovne:

```
NodeContainer p2pNodes;

p2pNodes.Create (2);

PointToPointHelper pointToPoint;

pointToPoint.SetDeviceAttribute ("DataRate", StringValue ("5Mbps"));
pointToPoint.SetChannelAttribute ("Delay", StringValue ("2ms"));

NetDeviceContainer p2pDevices;

p2pDevices = pointToPoint.Install (p2pNodes);
```

Tu vytvárame 2 uzly a point-to-point konfiguráciu s vlastnými atribútami a následne ju inštalujeme na vytvorený kontajner.

Keďže vo VANET sieťach sa budeme zaoberať bezdrôtovými sieťami, tak sa bližšie pozrieme na simuláciu WIFI zariadení v ns-3. V základnej wifi konfigurácii sú 2 typy wifi uzlov. AP (access point) uzol a STA(station node). AP uzly sú stacionárne a STA uzly sa vedú pohybovať.

```
NodeContainer wifiStaNodes;

wifiStaNodes.Create (nWifi);

NodeContainer wifiApNode = p2pNodes.Get (0);
```

Následne je potrebné nakonfigurovať jednotlivé vrstvy - aspoň fyzickú a mac vrstvu. Pri fyzickej sa nastavuje aký kanál bude bezdrôtová komunikácia používať. NS-3 poskytuje aj základný kanál, ktorý je možno použiť.


```

YansWifiChannelHelper channel = YansWifiChannelHelper::Default ();

YansWifiPhyHelper phy = YansWifiPhyHelper::Default ();

phy.SetChannel (channel.Create ());

```

Na mac vrstve sa okrem iného dá nastaviť SSID siete, typ wifi uzla - AP alebo STA a či má aktívne vyhľadávať nové siete v jeho rozsahu.

```

WifiMacHelper mac;

Ssid ssid = Ssid ("ns-3-ssid");

mac.SetType ("ns3::StaWifiMac",

             "Ssid", SsidValue (ssid),

             "ActiveProbing", BooleanValue (false));

```

Pri bezdrôtových sieťach je taktiež potrebné zadať fyzickú polohu uzlu, keďže vzdialenosti majú vplyv napríklad na signál. NS-3 na to poskytuje triedu MobilityHelper, vďaka ktorej vieme namapovať uzly na virtuálny 2D priestor, ktorého rozsah vieme špecifikovať.

```

mobility.SetPositionAllocator ("ns3::GridPositionAllocator",

                               "MinX", DoubleValue (0.0),

                               "MinY", DoubleValue (0.0),

                               "DeltaX", DoubleValue (5.0),

                               "DeltaY", DoubleValue (10.0),

                               "GridWidth", UIntegerValue (3),

                               "LayoutType", StringValue ("RowFirst"));

```

Správanie jednotlivých uzlov vieme zadať cez mobilitymodel, napríklad aby bol uzol stacionárny, alebo sa náhodne pohyboval v určitom rozsahu

```

mobility.SetMobilityModel ("ns3::RandomWalk2dMobilityModel",

                           "Bounds", RectangleValue (Rectangle (-50,

50, -50, 50)));

```

```
mobility.Install (wifiStaNodes);
```

Ostatné typy bezdrôtových sietí budú analyzované v nasledujúcich šprintoch.

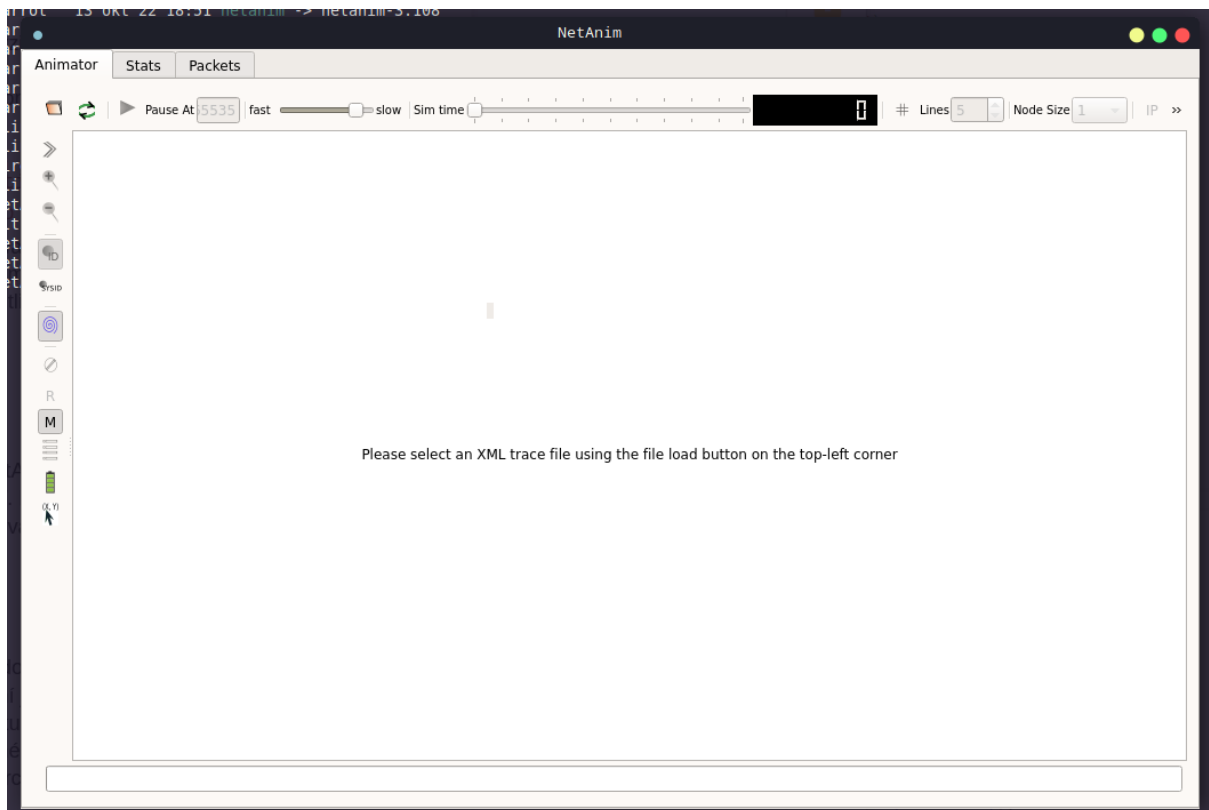
Aby bol scenár aspoň trochu modulárny ns-3 umožňuje konfiguráciu scenára cez command line argumenty. Slúži na to trieda CmmmandLine.

```
CommandLine cmd;  
  
cmd.AddValue ("verbose", "Tell echo applications to log", verbose);  
  
cmd.AddValue ("tracing", "Save simulation to trace files", tracing);
```

Program sa s argumentami spúšťa cez waf cez syntax `./waf -run "/scratch/scenar -argument=hodnota -argument2=hodnota2"`. Okrem informácii o logovaní zvykne byť týmto spôsobom konfigurovateľný napríklad počet uzlov v jednotlivých kontajneroch.

Vizualizácia výstupov simulácie

Pri inštalácii ns-3 dostaneme zdrojové kódy programu NetAnim, ktorý umožňuje vizualizáciu takejto simulácie. Zdrojové kódy treba najprv skompilovať. Je to nástroj s GUI, takže ho treba spúšťať na systéme s grafickým rozhraním, nie len terminálom. Keďže je aplikácia napísaná pomocou Qt knižnice, najjednoduchší spôsob je nainštalovať QtCreator, otvoriť v ňom súbor projektu NetAnim.pro a skompilovať ho cez QMake priamo tam.



Po spustení je vidieť, že NetAnim si pýta XML trace súbor. Na vytvorenie takéhoto súboru treba rozšíriť ns-3 simuláciu o pár riadkov kódu

```
AnimationInterface anim("third.xml");

for(uint32_t i=0; i<=nCsma; i++) {

    anim.SetConstantPosition(csmaNodes.Get(i), 10.0, 10.0*i*2);

}

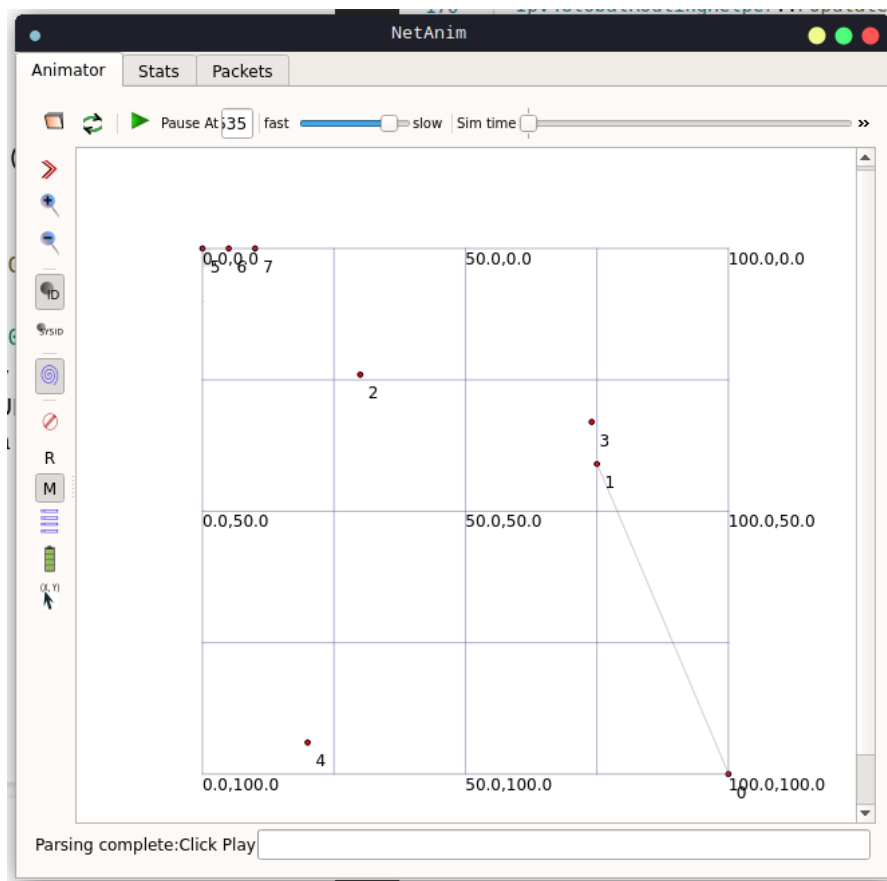
for(uint32_t i=0; i<nWifi; i++ ) {

    anim.SetConstantPosition(wifiStaNodes.Get(i), 10.0*2*i, 10.0);

}

anim.SetConstantPosition(wifiApNode.Get(0), 100.0, 100.0);
```

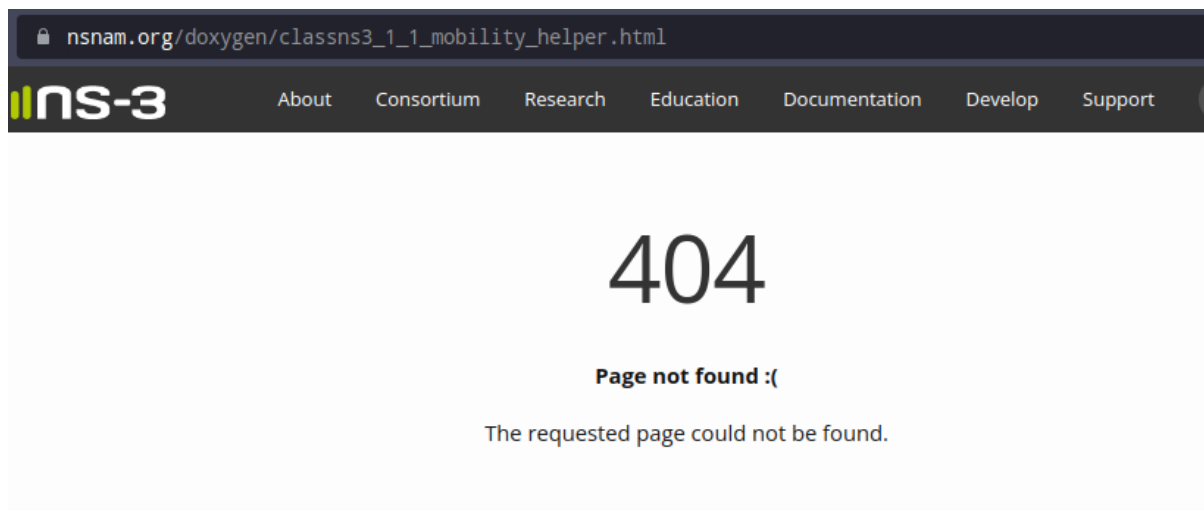
Hlavnou súčasťou kódu je objekt AnimationInterface, ktorý vytvorí/prepíše súbor v konštrukte a bude do neho vkladať logy. Keďže ide o GUI tool ktorý vizualizuje uzly v 2D priestore, tak každému uzlu musí byť pridelená pozícia, na ktorú sa vykreslí.



Počiatočná pozícia sa vykreslí a simulácia sa dá simulovať pomocou zeleného tlačidla v toolbare programu. V tabe Stats sa dajú pozrieť štatistiky a v tabe Packets odoslané pakety. Program je však labilný takže treba s ním narábať opatrne, inak segfaultuje.

Nedostatky

Bohužiaľ, ns3 je veľmi riedko zdokumentované. Kvalitná dokumentácia končí pri inštalácii, konfigurácii a spustení prerobených simulácií. Pri googlení je polovica zdrojov mŕtva a hádže 404 (obrázok) alebo je dokumentácia limitovaná na ukážku používania bez žiadneho opisu čo/prečo/ako. Najväčší zdroj informácií sú teda predrobené scenáre v priečinku examples a hŕstka videí uploadnutých na youtube od nezávislých tvorcov čo dané scenáre vysvetľujú.

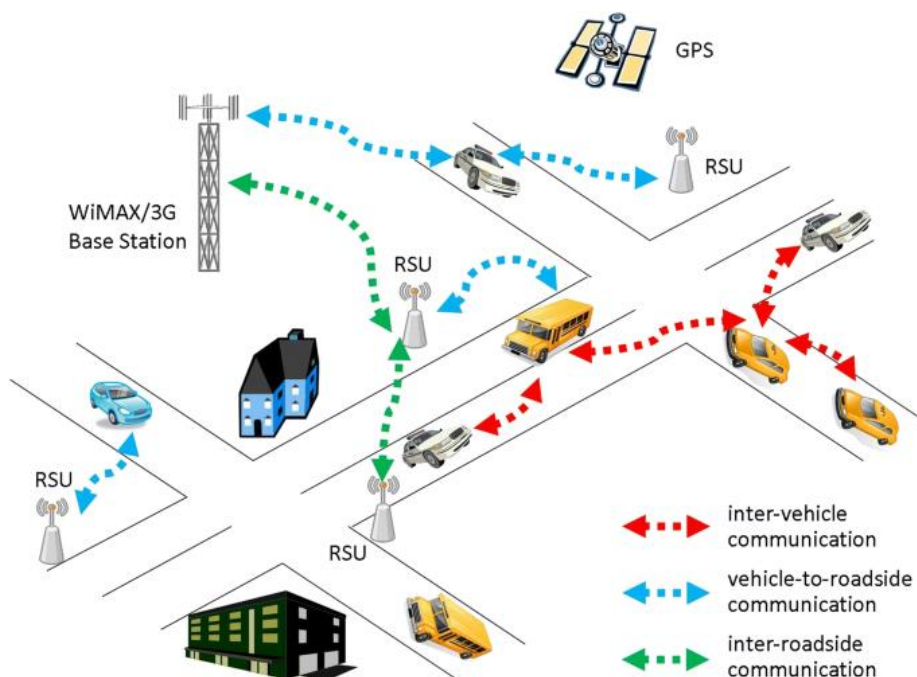


Zdroje

<https://www.nsnam.org/wiki/Installation>

3.4 VANET

Vehicular ad hoc network alebo VANET je rýchlo rastúca technológia zaoberajúca sa komunikáciou na cestách medzi vozidlami navzájom a okolitým priestorom. Prvé VANET siete vznikli v roku 2001 s účelom zvýšiť bezpečnosť na cestách a zjednodušiť navigáciu. Technológia VANET je malá časť technológie MANET (Mobile ad hoc network), teda pohyblivé siete. Pri technológii VANET sa používajú ako jednotlivé pohyblivé uzly siete autá. Autá pripojené do VANET siete sú schopné komunikovať medzi sebou aj s RSU (Roadside Units), teda so statickými vysielačmi pri ceste. VANET komunikácie sa delia na v2v (vehicle to vehicle alebo komunikácia medzi vozidlami) a v2i (vehicle to infrastructure alebo komunikácia medzi vozidlom a okolitou infraštruktúrou) komunikáciu.



Využitie VANET sietí

Jedno z prvých využití VANET siete je ad-hoc komunitizácia áut. V tejto funkcii VANET sieť identifikuje skupinu áut pripojených na VANET ako komunitu s rovnakým cieľom a smerom cesty. Keďže tieto autá idú na to isté miesto, dá sa predpokladať, že v určitom bode cesty sa stretnú a pôjdu spolu. Pomocou tejto funkcie sa dá lepšie plánovať cesty pre skupiny áut.

Ďalšie využitie VANETu je tzv. „platooning“. Pri platooningu sú schopné autá nasledovať za sebou s veľmi malými medzerami. Pomocou VANET si autá neustále vymieňajú informácie o rýchlosti, akcelerácii a zabáčaní.

VANET má ešte mnoho ďalších využití ako napríklad: elektronické brzdné svetlo v prípade keď auto vpredu nie je vidno, varovania pred kolíziou, pomoc pri zmene pruhov, hľadanie strateného alebo kradnutého vozidla, internetové pripojenie a mnohé ďalšie.

V2V a V2I

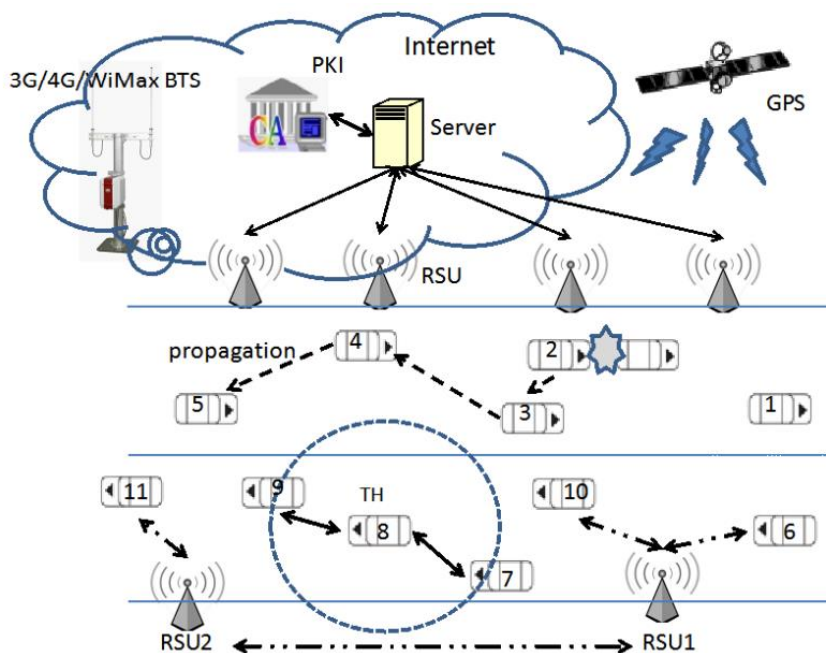
Keďže topológia VANET siete sa neustále mení, protokoly vo VANET delíme na dve skupiny: protokoly zložené na topológii siete a protokoly založené na polohe.

Topologické protokoly sledujú prepojenia a uzly v sieti, ktoré už existujú a pomocou nich posielajú dáta určeným uzlom. Proaktívne topologické protokoly využívajú a udržiavajú exkluzívne už vytvorené prepojenia, čo môže byť energeticky náročné pre často meniace sa VANET siete. Reaktívne topologické protokoly vytvárajú cestu k cieľu, len keď ju

potrebujú a po komunikácii neudržia spojenie. Preto pri tejto komunikácii vznikne zdržanie, keď sa na chvíľu preruší. Najčastejšie používané, hybrid topologické protokoly kombinujú proaktívne a reaktívne protokoly a používajú ich výhody tam, kde treba. Vďaka nim je VANET sieť efektívna a vysoko škálovateľná.

Topology Based Routing Protocols	Position Based Routing Protocols
Need of route maintenance for all routes.	No need of route maintenance.
Require large bandwidth if network topology changes.	Does not require large bandwidth.
Forwarding decision is based on the source node.	Forwarding decision is based on the position of destination and the next hop neighbor.
Based on route discovery scheme.	Based on location service scheme.
DSDV, OLSR, AODV, DSR, TORA, ZRP, etc.	GPSR, A-STAR, AMAR, GyTAR, EBGR, MFR, B-MFR, etc.

Polohové protokoly potrebujú k správnej funkcionalite presnú fyzickú pozíciu uzla v sieti. Táto pozícia sa dá získať pomocou informácie poslanej pomocou vedľajších uzlov a topologických protokolov. Polohové protokoly sú užitočnejšie vo VANET sieťach, keďže sa dá predpokladať, že pohyblivé uzly, teda autá sa budú hýbať po ceste, takže sa im dá aj predpovedať pozícia podľa aktuálnej rýchlosti. Vo VANET sieti cesta je niekoľko vozidiel ktorým môžeme predikovať polohu podľa informácii o ich smere a rýchlosti, čo si navzájom posielajú. Tu je veľmi užitočná funkcionalita zoskupovania áut podľa cesty.



Pri V2I vozidlá sa pripoja do VANET pomocou RSU, teda statickými vysielačmi blízko cesty. Keďže je prakticky nemožné lemoviť týmito vysielačmi každú jednu cestu, VANET siete sa hlavne spoliehajú na V2V komunikáciu. RSU vysielače existujú aby dokázali jednoduchšie prepojiť vozidlá vo VANET sieti aj aby ich spojili s VANET serverom a internetom.

RSU dokážu lepšie dohliadnuť na a monitorovať zoskupenia áut

Záver

VANET technológia je aktuálny „ďalší krok“ k samojazdiacim autám. Aktuálne nie sú dokonalé a ich funkcionality nie je ešte úplne vyvinutá. Aktuálne väčšina „smart“ aut na ceste používa technológiu VANET ale stále ich nie je dosť na úplnú efektivitu siete. Postupom času, čím viac „smart“ vozidiel pribudne na cestách, tým viac budú VANET siete užitočnejšie.

Zdroje

<https://www.sciencedirect.com/science/article/pii/S2214209620300814>

<https://www.tandfonline.com/doi/full/10.1080/23311916.2017.1362802>

https://www.researchgate.net/figure/An-example-of-a-VANET_fig1_264815864

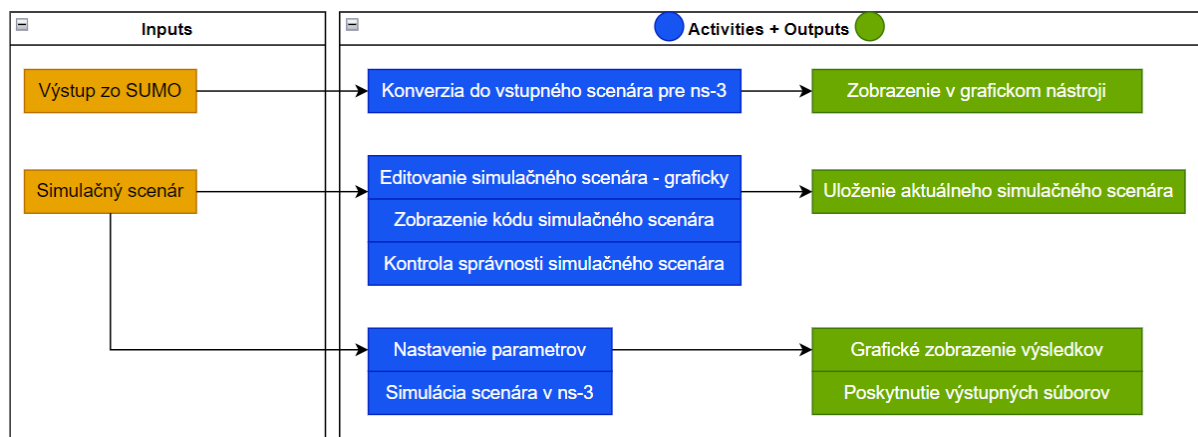
<https://www.sciencedirect.com/topics/computer-science/vehicular-ad-hoc-network>

<https://encyclopedia.pub/entry/8743>

https://www.researchgate.net/publication/50392063_Performance_comparison_of_position-based_routing_protocols_in_vehicle-to-vehicle_v2v_communication

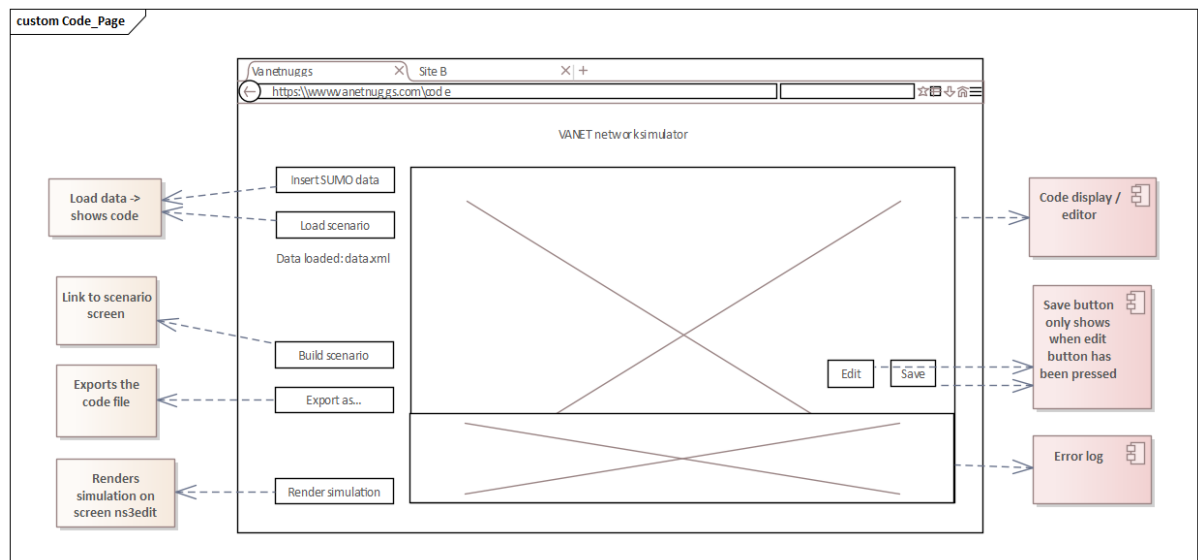
4 NÁVRH APLIKÁCIE

4.1 Logický model aplikácie



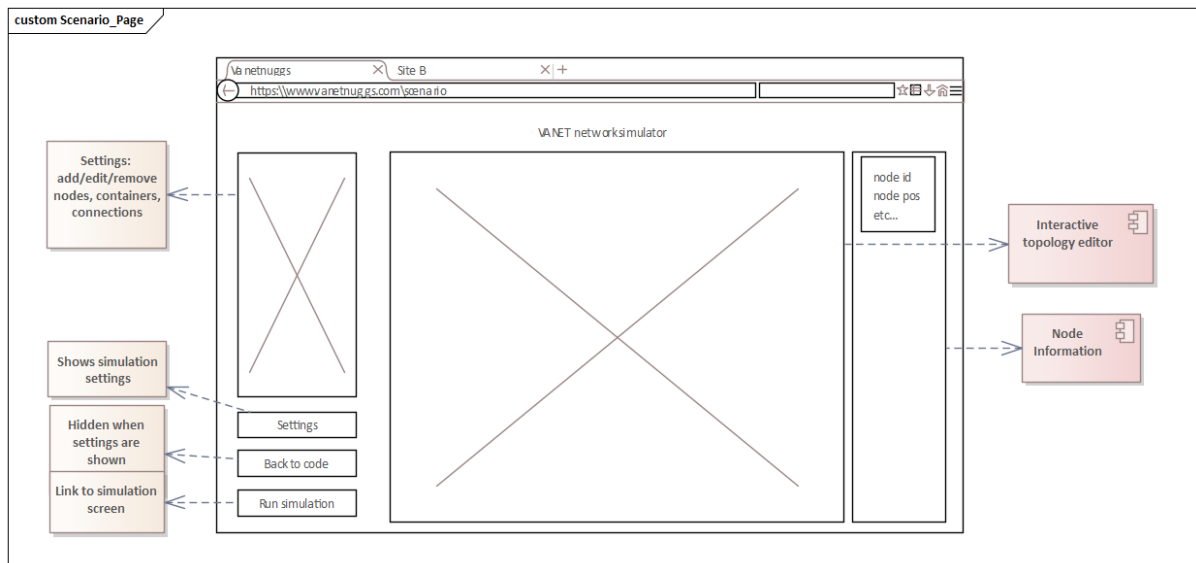
4.2 Drôtené modely

Code page



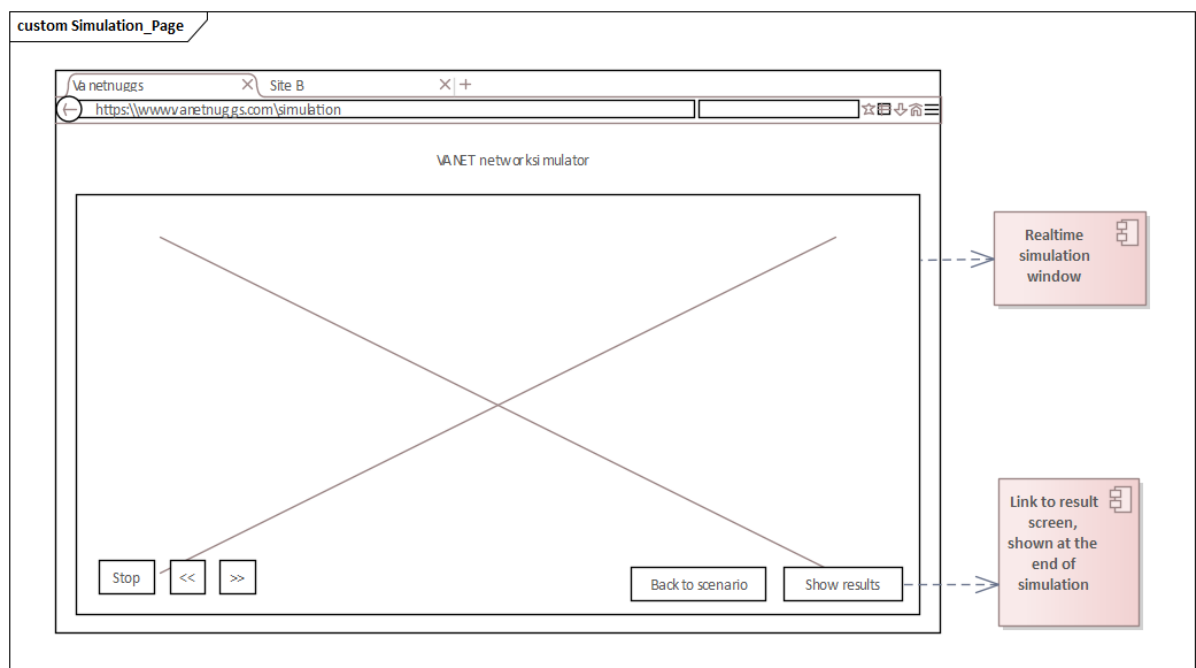
Code page je prvá stránka, ktorú používateľ uvidí. V nej môže nahráť vlastný python kód alebo scenár a editovať ho. Po edite môže kód exportovať alebo spustiť simuláciu.

Scenario page



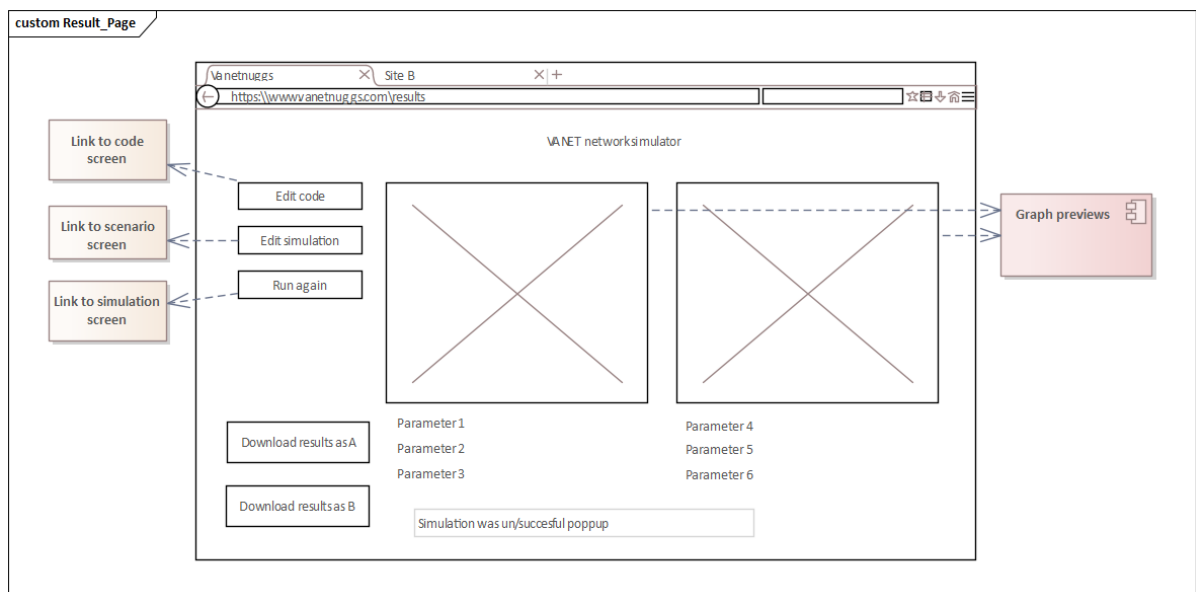
V scenario page môžeme editovať scenár pomocou interaktívnej obrazovky. Môžeme tu pridávať kontajnery a nody a meniť ich nastavenia. Po editovaní môžeme spustiť simuláciu.

Simulation page



Na tejto obrazovke vidíme priebeh simulácie. Ak je používateľ nespokojný, môže ísť späť na scenario page. Po dokončení môžeme prejsť na výsledkovú obrazovku.

Result page



Toto je finálna obrazovka aplikácie. Jej hlavná úloha je zobrazíť namerané dáta zo simulácie. Výsledky sa potom dajú aj stiahnuť. Dá sa z nej vrátiť na akúkoľvek inú obrazovku.

5 PROTOTYP APLIKÁCIE

Prototyp bude vedieť zostaviť a simulovať topológiou typu [second.cc](#) a [second.py](#). Z topologického hľadiska to sú uzly typu CSMA, Point-to-point a wifi. Komunikácia medzi uzlami bude zabezpečená UDP echo klientom a serverom.

Prototyp bude vedieť zostaviť topológiu s ľubovoľným počtom uzlov, klientov a serverov, ktorá sa bude dať odsimulovať.

5.1 Vybrané technológie

Front-end je implementovaný v jazyku svelte, oproti iným webovým frameworkom je rýchlejší, nakoľko je kompilovaný a po builde neobsahuje zbytočnosti. Back-end je implementovaný v jazyku python s frameworkom Flask.

5.2 Front-end

V NS3 simuláciach sa konfigurácie inštalujú buď na uzly, alebo na kontajnery, ktoré obsahujú viacero uzlov s rovnakou konfiguráciou. Z tohoto dôvodu bude implementácia uzlov zjednotená do kontajnerov. Pre prehľadnosť a reaktivitu sú všetky vlastnosti konfigurácie uložené v globálnom svelte store.

Rozhranie front-endu našej aplikácie je rozdelené na 5 častí - horný a dolný toolbar, ľavý a pravý drawer, a nakoniec hlavný kontend ktorý sa nachádza v strede obrazovky. Horný a dolný toolbar slúžia najmä na komunikáciu z backendom a zobrazovanie vrátenej výslednej konfigurácie. Na dolnom toolbare sa nachádza tlačidlo "**Send scenario**". Toto tlačidlo posiela scenáre nakonfigurované z front-endu na back-end. Odpoveď sa dá následne zobrazíť cez tlačidlo "**summary DragDrop**", ktoré sa nachádza v hornom toolbare.

Ďalšími dôležitými časťami front-endu sú pravý a ľavý drawer. Pravý drawer nám bližšie popisuje, čo sa nachádza na obrazovke v sekcii hlavný kontent. Tu sa ukazujú informácie o aktuálnom node. To znamená posledný node, na ktorý používateľ ukázal. Nachádza sa tu informácia o id nodu, o jeho realnej polohe v priestore a o aký typ nodu ide. Tento typ je modifikovateľný. Rovnako sú tu spísané aj jednotlivé kontainery, v ktorých sa node nachádza. Ak neexistuje aktuálny node, čo znamená, že používateľ na žiaden neukázal, tento priestor je prázdny.

Je rozdelený na 2 časti. Prvá časť sa stará o kontajnere, teda jednotlivé siete. Umožňuje vytvárať vytvárať rôzne typy sietí. Zatiaľ sú implementované 3 typy

- P2P, csma, wifi. V týchto kontajneroch vie užívateľ nastavovať ich atribúty, ako je IP adresa, maska siete a časové intervaly potrebné pre simuláciu. V každej sieti môžeme pridávať nody, ktoré v nej medzi sebou komunikujú - v P2P len 2, v csma a wifi ľubovoľný počet. Vo wifi je potrebné zvoliť, ktorý node bude reprezentovať AP.

Je možné vytvárať ľubovoľné množstvo kontajnerov a taktiež ich aj mazať. Druhá časť draweru v sebe obsahuje aplikácie, ktoré budú simulované. Aplikácie sú rozdelené na 2 typy - serverové a klientske. Pri oboch je potrebné zadať časový interval, odkedy bežia, na akom porte sú spustené a kontajner - sieť - v ktorom bežia. Serverová aplikácia potrebuje špecifikovať, na ktorom node beží. Klientskej aplikácii je zase potrebné špecifikovať, na ktorú serverovú aplikáciu sa pripája. Okrem toho sa tu zadávajú údaje o packetoch, ako ich počet a maximálna veľkosť.

V hlavnom kontexte je samotná Drag and Drop časť aplikácie. Tu používateľ môže vytvárať a jednoduchým pohybom myšky aj posúvať jednotlivé nody. Každý nod predstavuje nejaký objekt, ktorý obsahuje svoje ID a červený krížik, ktorý slúži na odstránenie tohto nodu. V prípade, že sú jednotlivé nody vo vzájomnom vzťahu, táto skutočnosť je prezentovaná čiarami, ktoré ich spájajú. Ak sa niektoré nody nachádzajú vo viacerých kontajneroch naraz, ich vzťah je reprezentovaný jednou čiarou, nad ktorou sú vypísané mená týchto kontajnerov. Tento spôsob bol zvolený pre lepšiu viditeľnosť. V tejto časti bol implementovaný aj zoom. Funguje na točenie kolieska na miške. Maximálny zoom je 3x a najmenší je 0.3x. Ak však používateľ chce túto funkčnosť vypnúť, môže tak spraviť tlačidlom "**Fix to 1x**", kde budú pozície a veľkosti jednotlivých nodov vrátené do pôvodnej pozície.

Hlavná scéna má 2 režimy - jeden na interaktívne vytváranie topológie a druhý na zobrazenie výsledkov simulácie.

Prepínať medzi režimmi sa dá cez tlačidlá v hornom toolbare.

5.3 Kontajner a jeho atribúty

- Unikátny názov kontajneru
- Typ kontajneru (CSMA/P2P)
- Atribúty kontajneru - zatiaľ spoločné pre obidva typy kontajneru
- Delay - konfiguračná hodnota - oneskorenie odoslania správy
- DataRate - konfiguračná hodnota - priepustnosť spojenia
- Adresa siete - IP adresa siete, z ktorej rozsahu sa budú nastavovať adresy jednotlivým uzlom v kontajneri • Maska siete - Maska siete vyššie špecifikovanej adresy
- Názov siete - bude odstránené, nepotrebné.

Do kontajnerov sa budú môcť vkladať uzly. Každý uzol môže byť v ľubovoľnom počte kontajnerov, to že je uzol v kontajneri si môžeme predstaviť ako sieťové interface uzlu.

5.4 Aplikácie

1. UDP echo server

Ide o službu, ktorá počúva na porte sieťového rozhrania a odpovedá na UDP echo požiadavky. Na jeho vytvorenie potrebuje atribúty:

- Názov serveru
- Čas spustenia
- Čas vypnutia
- Port, na ktorom bude počúvať
- Kontajner, z ktorého uzlov sa bude vyberať uzol na nainštalovanie
- ID uzla na ktorý bude server nainštalovaný (jeden zo zvoleného kontajneru)

2. UDP echo klient

Ide o službu, ktorá bude posilať požiadavky na zvolený server a bude počúvať odpovede. Má atribúty:

- server, na ktorý bude posilať požiadavky
- čas spustenia
- čas vypnutia
- kontajner, v ktorom bude klient nainštalovaný
- ID uzla v rámci kontajnera, na ktorý bude klient nainštalovaný
- konfiguračné atribúty: - interval odosielania - maxialny počet odoslaných paketov - veľkosť individuálnych paketov

Konfigurácia je implementovaná cez HTML formuláry a pridávanie a odoberanie uzlov cez grafické drag-and-drop interaktívne rozhranie. Spojenia medzi uzlami v rámci kontajneru budú dynamicky graficky zobrazené.

Aplikácia vytvorí veľký JSON obsahujúci celú konfiguráciu scenára, ktorá bude sparsovaná a odsimulovaná na back- ende. Výstup sa bude vedieť vizualizovať jednoduchou formou zobrazenia správ .pcap súbory. Príklad vygenerovaného JSONu:


```

{
  "topology": {
    "node_count": 6,
    "node_containers": [
      "csma_nodes",
      "p2p_nodes"
    ],
    "container_settings": {
      "p2p_nodes": {
        "id": 1,
        "name": "p2p_nodes",
        "type": "point_to_point",
        "data_rate": {
          "value": 100,
          "format": "Mbps"
        },
        "delay": {
          "value": 1,
          "format": "ns"
        },
        "network_name": "p2p_interfaces",
        "network_address": "10.2.2.0",
        "network_mask": "255.255.255.0",
        "log_pcap": true,
        "nodes": [
          7, 9
        ]
      },
      "csma_nodes": {
        "id": 0,
        "name": "csma_nodes_interfaces",
        "type": "csma",
        "data_rate": {
          "value": 100,
          "format": "Mbps"
        },
        "delay": {
          "value": 1,
          "format": "ns"
        },
        "network_name": "csma_interfaces",
        "network_address": "10.1.1.0",
        "network_mask": "255.255.255.0",
        "log_pcap": true,
        "nodes": [
          2, 3, 4, 7
        ]
      }
    }
  },
  "simulation": {
    "server": {
      "echo_server": {
        "id": 0,
        "name": "echo_server",
        "port": 9,
        "start": {
          "value": 1,
          "format": "s"
        },
        "stop": {
          "value": 10,
          "format": "s"
        },
        "network": "csma_nodes",
        "node": 7
      }
    },
    "client": {
      "echo_client": {
        "id": 0,
        "name": "echo_client",
        "port": 9,
        "start": {
          "value": 1,
          "format": "s"
        },
        "stop": {
          "value": 10,
          "format": "s"
        },
        "network": "csma_nodes",
        "node": 2,
        "server": "echo_server",
        "interval": {
          "value": 1,
          "format": "s"
        },
        "max_packets": 10,
        "packet_size": 128
      }
    }
  }
}

```

5.5 Back-end

Backend slúži na parsovanie JSONu topológie a jej transformáciu na simulačný scenár. Poskytuje dva API endpointy.

POST /tracejson - V tele požiadavky dostane JSON scenáru v ygenerovaný front-end časťou aplikácie, ktorý sa pretransformuje na simulačný scenár, spustí v NS-3 simulátore a vráti výstup simulácie v JSON objekte. Výstupný objekt obsahuje dva atribúty:

- logs: list<{string: name, int: size}> - zoznam vygenerovaných pcap súborov - dvojica názov súboru a jeho veľkosť
- output: list<string> - zoznam výstupu scenáru. Správy typu uzol x poslal y bajtov na IP adresu z .

GET /trace?name=XYZ - Ako argument name dostane meno .pcap súboru, získané cez /tracejson požiadavku. Vráti .pcap súbor s rovnakým meno.

Po prijatí JSON-u sa zavolá parser. Ten je zložený z viacerých modulov. Pre prehľadnosť každý modul parsuje logický celok scenáru. Aktuálne aplikácia podporuje parsovanie nasledujúcich celkov cez moduly:

- nodes - rozdelenie uzlov do kontajnerov
- p2p - konfigurácia point-to-point kontajnerov
- csma - konfigurácia csma kontajnerov
- echoudp - echo udp klient/server-y
- ipv4 - ipv4 komunikácia a adresy
- log - logovanie - aké ns3 udalosti sa majú logovať, zatiaľ nastavené staticky.
- pcap - vytváranie pcap súborov pre komunikáciu v rámci kontajnerov

Vytiahnuté údaje z JSON objektu sú presmerované do virtuálneho modelu jednotlivých prvkov scenára. Momentálne je model implementovaný len pre zložitejšie prvky scenáru - udp klient a server. Virtuálny model má okrem konštruktoru implementované funkcie

- dumpypy - vráti python zdrojový kód simulačného scenáru modelu
- dumpc++ - zatiaľ neimplementované - to isté ale pre jazyk c++.

Na formátovanie elementov ako čas a formát atribútov uzlov slúžia takzvaní helperi, máme implementovaný jeden, `format_helper`, ktorý parsuje údaje na formát, ktorý NS-3 potrebuje.

Parser zavolá všetky moduly a vytvorí zdrojový kód scenáru, ktorý sa uloží na disk a vykoná v ns-3 simulácii.

API endpoint má 2 pomocné manažéry

- File manažér, ktorý má na starosti prácu so súborovým systémom - ukladať vytvorené scenáre na disk, spustenie scenárov, kopírovanie výstupných súborov scenáru do priečinku dostupného cez API.
- NS-3 manažér, ktorý má na starosti operácie s NS-3 simulátorom - vykonanie scenáru ak pozná názvy súboru, získanie vygenerovaných logov a podobne.

V tomto prototype backend poskytuje aj jednoduchú statickú web-stránku pod endpointom `GET /`, ktorá slúži na testovanie. V neskorších verziách programu bude odstránená, nakoľko nebude potrebná.

5.6 Príručka užívateľa

Tento projekt je možné spustiť pomocou dockeru alebo rovno na stránke, čo je jednoduchší variant bez nutnosti nejakej inštalácie. Otvoríme si projekt na stránke: <https://vanetlab.ml/>. Po otvorení môžeme vidieť 3 hlavné časti:

Lavý drawer

Topology settings

Možnosť na vytváranie kontajnerov do, ktorých následne vieme pridávať jednotlivé nody. Najprv zadáme meno kontrainera a potom vyberieme typ z aktuálne podporovaných. Ak nezadáme žiadne meno automaticky sa nám vygeneruje.

Po kliknutí tlačidla “create” sa nám pridá kontajner do listu. Takto môžeme vidieť pod sebou vytvorené kontajnery. Kontajnery sa dajú vymazať tlačidlom “x”, ktoré sa nachádza vedľa každého kontajnera.

Ak klikneme na meno kontajnera, otvorí sa nám konfiguračné menu. V tomto menu sa dajú nastavovať jednotlivé metriky pre daný

kontajner ako aj priradené nody, ktoré vieme zobraziť tlačidlom “show nodes”. Nody pridávame/odoberáme kliknutím na tick box. Označené nody sú priradené danému kontajneru.

Simulation settings

V podstate platí to isté ako pri topology settings so smenou, že sa jedná o nastavenia aplikácie. Buď klientskej alebo serverovej časti. Nastavenie vieme pridávať tlačidlom “add” a po pridaní je možné nakonfigurovať jednotlivé parametre. (WIP funkcionálnosť)

Presets

Na ukážku máme vytvorené 2 presety, csma a wifi. Po kliknutí na preset sa nám zobrazia nody na plátno, kde s nimi vieme interagovať. Presety je vhodné nemodifikovať, pretože sa nemusia potom správať správne.

Canvas

Jedná sa o obrazovku, na ktorej sa zobrazujú nody a ich prepojenia. Tlačidlom „add node“ pridávame nody, ktoré je potom možné pridávať do kontajnerov ako už bolo spomenuté v predchádzajúcej časti. Medzi nodami spadajúce pod jeden kontajner sú vykreslené prepojenia. Tieto prepojenia sú farebne odlišené a každý kontajner má unikátnu farbu. Ak node spadá do viac ako jedného kontajnera, nad prepojením sa zobrazia mená všetkých kontajnerov v ktorých sa nachádza.

Nody vieme zmazať kliknutím na tlačidlo „X“. Po odstránení sa node vymaže aj s kontajnerom spolu s prepojeniami. Ak chceme nody posúvať, klikneme na druhé tlačidlo a potom na node. Označený node je potom zvýraznený červenou farbou. Takýmto spôsobom vieme označiť viacero nodov naraz ak držíme tlačidlo „shift“. Ak chceme posunúť mapu, zložíme tlačidlo mapy a myšou posunieme plátno.

Canvas ponúka aj možnosť približovania o, ktorú sa starajú tlačidlá na pravej strane. Ak chceme priblíženie resetovať, použijeme tlačidlo reset.

Pravý drawer

Ak chceme vedieť detailnejšie informácie o konkrétnom node, klikneme na štvrté tlačidlo s ikonkou „i“ a potom na node. V pravom drawere sa nám zobrazí detailné informácie o vybranom node.

Top burger menu

V hornom burger menu si vieme vybrať medzi canvas a simulation.

Canvas je už spomínané vyššie. Simulation nám ponúka vidieť informácie o jednotlivých scenároch. Ak sa nám najprv nezobrazia žiadne scenáre, je potrebné stlačiť „refresh“. Pri každom scenári si vieme pozrieť logy, výstup z konzole ako aj vygenerovaný source code v jazyku python. Po kliknutí na logy si vieme stiahnuť aj jednotlivé pcap súbory.

6 ŠPRINTY

6.1 Šprint 1 – Operation Homemasters

Zhrnutie sprintu:

Zhrnutie:

Meno	Úlohy	Mid-Sprint review	MS%	EOS review	ES%
Richard Andrášik (Scrum Master)	Analýza VANET technológií	PN prvý týždeň, žiaden postup	0%	Analýza hotová, kvalitné, stručné spracovanie	100%
Samuel Kačeriak	Analýza SDN, tmavý mód a spodný komponent pre webstránku	Komponenty hotové, SDN nie je začatý	80%	Dokumentácia SDN dokončená, do hĺbky vypracované	100%
Andrej Dubovský	Analýza SUMO a komponent na dokumenty	Všetko hotové	100%	Stále všetko hotové	100%
Roman Grom	Vedenie analýzy SUMO a inštalácia	SUMO nainštalované, treba sa na to viac pozrieť	80%	Naučil sa ako SUMO funguje	100%
Ondrej Martinka	Množstvo komponentov pre stránku, analýza a inštalácia ns-3	Šikovne porobil množstvo funkcií do webstránky, ns-3 analýza ešte nie je	60%	Analýza ns-3 úplne dokončená, naštudované aj alternatívy	100%
Vojtech Fudaly	Výplň webstránky, srdce tímu	Srdce tímu na 100% Stránka vyzerá pekne	70%	Dokončený zvyšok komponentov	100%

- Trvanie sprintu: 18.10 – 4.11

V tomto sprinte sme vykonali množstvo dôležitej analýzy ohľadom technológií potrebných na prácu na projekte. Okrem toho sme vytvorili domovskú stránku tímu a zoznámili sa s agile pracovným postupom.

Začiatok sprintu:

Cieľom prvého, testovacieho sprintu je navedenie sa do agile pracovného štýlu a otestovanie si agile praktík. Úlohy v prvom sprinte boli vytvorenie timovej stránky analýza technológií, ktoré budú v projekte

používané a vytvorenie organizácie na GitHubu, kde budeme verziovať náš kód. Na druhom stretnutí tímového projektu, boli rozdelené všetky úlohy. Tento sprint trval štandardný čas, teda dva týždne.

Rekapitulácia v strede sprintu:

V strede sprintu sme prišli hlavne na to že rozdelených úloh bolo málo, keďže viacerí z nás by dokázali všetku pridelenú prácu urobiť za deň. Za prvý týždeň šprintu sme urobili viac práce na webstránke a menej analýzy, lebo to sa nikomu z nás nechce. V strede sprintu je približne 60-70% práce hotová.

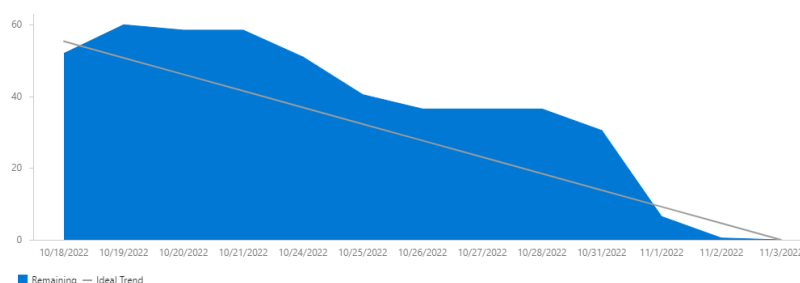
Reflexia sprintu

Ondrej efektívne analyzoval ns-3 a odovzdal výbornú ns3 dokumentáciu a urobil základ a množstvo komponentov tímovej stránky. Samo dokázal efektívne urobiť dokumentáciu o SDN a urobil parádny spodný komponent pre stránku. Roman nainštaloval SUMO a dokázal spustiť a rozumieť testovacej simulácii v SUMOe. Andrej parádne analyzoval SUMO, čo pomohlo Romanovej práci a pridal komponenty na vkladanie súborov na tímovú stránku. Vojto dokončil jemu priradené komponenty na tímovú stránku a teraz stránka vyzerá bezchybne. Rišo urobil dokonalú 10/10 dokumentáciu o VANET technológii a spravoval parádne tím ako pravý scrum master.

Prvý sprint bol dokončený úspešne, všetky úlohy boli splnené včas. Z organizačných dôvodov sme boli nútení presunúť koniec šprintu o dva dni. V rámci reflexie sme si uvedomili, že množstvo zadaných úloh bolo príliš nízke. V druhom šprinte zadáme väčšie množstvo úloh na každého člena.

V rámci sprintu bola úspešne urobená začiatková analýza potrebná pre začatie projektu. Taktiež sa nám podarilo vytvoriť parádnu tímovú stránku. Počas prvého sprintu sme si medzi sebou teoretické znalosti potrebné k rozvoju projektu. Do budúceho sprintu je potrebné určiť, ako bude aplikácia vyzeráť. Bude potrebné vytvoriť wireframery a endpointy aplikácie, ktorých sa môžeme držať pri vývoji. Okrem toho bude potrebné vytvoriť minimálne základy pre aplikáciu, do ktorých budeme dorábať.

Burndown graf:



6.2 Šprint 2 – Operation Bonfire

Zhrnutie sprintu:

Meno	Úlohy	Mid-Sprint review	MS%	EOS review	ES%
Richard Andrášik (Scrum Master)	Formality, zápisnice a návrh aplikácie	Zjednotil reflexie a sprinty aby dobre vyzerali	40%	Urobil metodiku a wireframe aplikácie	100%
Samuel Kačeriak	Frontend upratovanie, zdravý rozum	Málo dokončenej práce, inštalácia ns-3	10%	Stále málo dokončenej práce, žiadne pridelené úlohy	20%
Andrej Dubovský	Vytváranie ns-3 editora	Pracoval na stránke, páči sa mu sprint, keďže tu je osobne	50%	Urobil mapku, kde môže posúvať cestičky	100%
Roman Grom	Analýza funkcionality aplikácie a práca na návrhu aplikácie	Zatiaľ nemal pridelenú prácu	0%	Pomoc pri analýze funkcionality aplikácie, logický model	100%
Ondrej Martinka	Backend aplikácie: generovanie pcap súborov	Vytvoril funkčný prototyp časti backendu	60%	Dokončenie funkcionality na backende	100%
Vojtech Fudaly	Opraviť chyby po Andrejovi, návrh frameworku	Opravoval chyby na tímovej stránke, mobilná stránka	30%	Opravovanie chýb, pridávanie frontend komponentov	100%

Zhrnutie:

- Trvanie sprintu: 4.11 – 15.11

Dokončenie a oprava komponentov na tímovej stránke. Položili sme základy frontendu a backendu Vanet simulátora. Urobili sme analýzu metódy drag and drop, ktorá končila neúspechom. Zjednotili sme výzor našich zápisníkov a reflexii sprintov a položili základ metodikám.

Začiatok sprintu:

V druhom sprinte je potrebné položiť minimálne základy aplikácie. V tomto sprinte vytvoríme drôtové a logické modely aplikácie, podľa ktorých

vytvoríme základnú štruktúru aplikácie. Dohodli sme sa, že úlohy v tomto sprinte budú iba voľne rozdelené a členovia tímu si ich budú prideľovať podľa potreby. Tento sprint potrvá len 12 dní, lebo sme kvôli voľnu začali neskôr.

Rekapitulácia v strede sprintu:

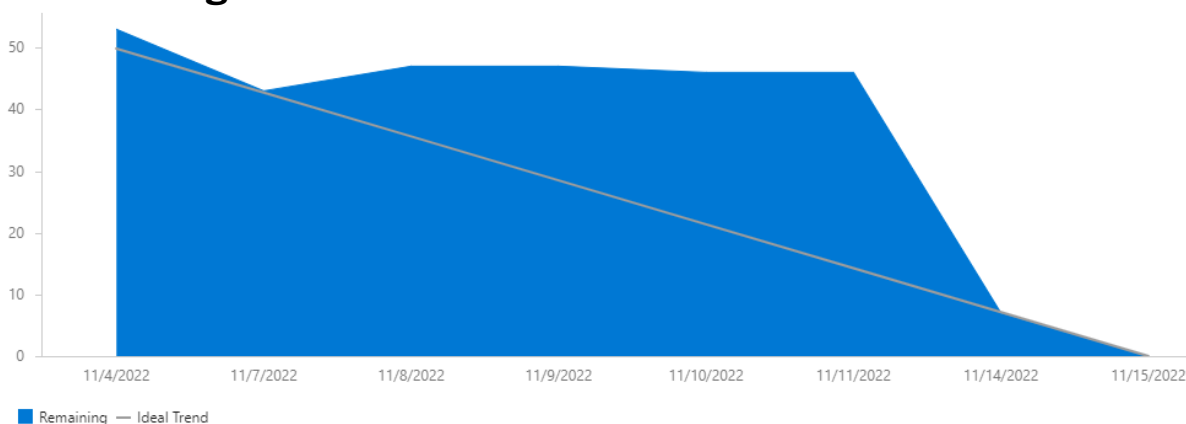
Meeting v strede sprintu sme mali len 4 dni po začatí sprintu, takže členovia tímu neurobili príliš veľa. V porovnaní s prvým sprintom, v tomto sprinte je už úloh akurát. Jednoduchý backend aplikácie je už hotový. Na konci stretnutia sme si rozdelili zvyšok úloh aj kto bude na projekte, čo robiť. Ondrej sa rozhodol robiť backend a Samo, Vojtech a Andrej budú pracovať na frontende. Richard a Roman urobia návrh a drôtené modely aplikácie.

Reflexia sprintu:

Na konci sprintu máme stále niektoré tasky nedokončené. Môže za to niekoľko faktorov: po prvé sprint trval kratšie ako obyčajne, ale počet úloh mal reprezentovať malé množstvo času. Niektorý/í nemenovaný/í člen/ovia tímu pomáhal/i iným namiesto robení vlastných úloh. Taktiež v tomto sprinte Scrum Master aplikoval metódu, kde nechal členov tímu vyberať úlohy počas celej práce. Touto metódou zostali úlohy na konci nepridelené.

V nasledujúcom sprinte Scrum Master nechá všetkých nech si sami pridávajú úlohy iba prvý týždeň. Pokiaľ pri rekapitulácii v strede sprintu zostanú úlohy nepridelené, prideli ich sám tak ako zväži. Je potrebné dať aj viac pozor aby viacerí nerobili to isté a aby neboli prekrývajúce úlohy.

Burndown graf:



6.3 Šprint 3 – Operation Men At Work

Zhrnutie sprintu:

Meno	Úlohy	Mid-Sprint review	MS%	EOS review	ES%
Richard Andrášik (Scrum Master)	Vytvoriť nové wireframy celej aplikácie	2 zo 4 wireframov sú hotové	35%	Všetky wireframy hotové	100%
Samuel Kačeriak	Metodika a prepojenia medzi uzlami siete	Metodika vyzerá hnusne, šípky sú ok	50%	Ondrej mu vymazal šípky, ale je done	100%
Andrej Dubovský	Lepší drag and drop, vytvorenie storu	Nevie čo ďalej, je úplne stratený	70%	Dokončil svoju prácu ale sám nevie ako	100%
Roman Grom	Komponent pre odosielanie dát do backendu	Briefing hotový, ešte nezačal s prácou	10%	Komponent dokončovaný	100%
Ondrej Martinka	Oprava drag and dropu Andrejovi, backend	Docker integrácia hotová, komponenty pre drag and drop	50%	Rework frontendu hotový, zakomponovanie dockera a prepojenia	100%
Vojtech Fudaly	Tímová webstránka, komponenty na frontend	Väčšina práce hotová, stále je srdce tímu	70%	Tech support, opravovanie bugov	100%

Zhrnutie:

- Trvanie sprintu: 15.11 – 29.11

Vytvorenie funkčného prototypu frontendu aplikácie a prepojenie s backendom. Po tomto sprinte je základná funkcionálna aplikácia hotová. Nové wireframy, metodika a komponenty tímovej stránky.

Začiatok sprintu:

Cieľom tohto sprintu je vytvoriť prezentovateľnú verziu aplikácie, čo už vyzerá v pohode. Stále však zostáva veľa práce a Andrej a Richard stále netušia čo sa v tomto projekte robí. Sprint obsahuje menšie množstvo komplikovanejších úloh ako obvyčajne. Potrvá štandardných 14 dní a všetci máme už od začiatku rozde

lené nejaké úlohy, ktorými začneme hneď.

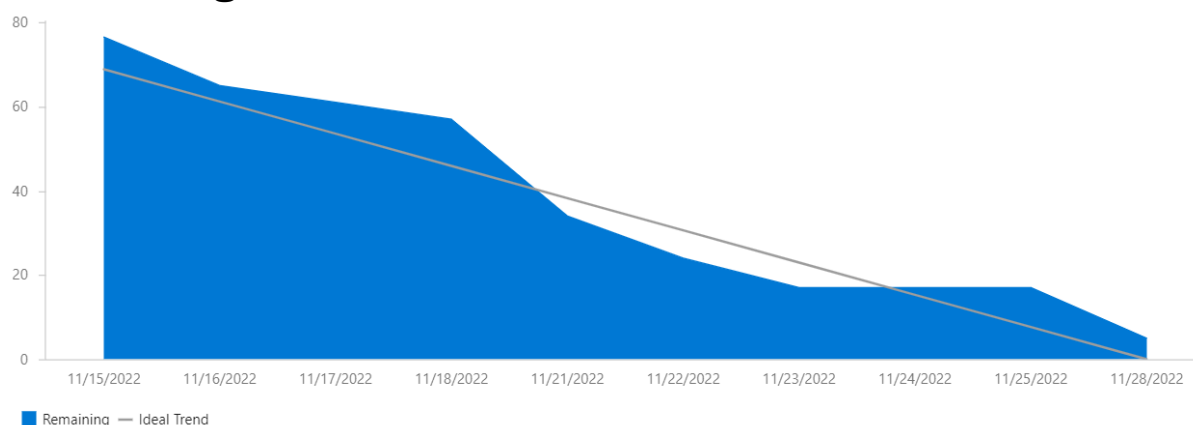
Rekapitulácia v strede sprintu:

Prvý krát sme počas sprintu pod priamkou ideálneho trendu. Máme ideálne množstvo úloh na dĺžku sprintu. Všetky úlohy sú rozdelené a pracuje sa na nich postupne, žiadne dodatočné pridelovanie nemusí scrum master v strede sprintu vykonávať. Väčšina úloh je už dokončená, mnohé sa práve vykonávajú a len málo úloh nebolo vôbec začatých. Týmto tempom skončíme s úlohami s miernym náskokom.

Reflexia sprintu:

Na konci sprintu sme v pohode hotový s úlohami. Niektorí z nás stihli urobiť úlohy v prvý týždeň a nemali prácu na druhý. Prvý krát sme skončili s náskokom aj keď máme problémy vymýšľať úlohy pre všetkých na ďalší sprint. V ďalšom sprinte plánujeme dať trochu viac úloh a prácu na aplikácie pre všetkých aby boli všetci zoznámení s kódom.

Burndown graf:



6.4 Šprint 4 – Corporate Espionage

Zhrnutie sprintu:

Meno	Úlohy	Mid-Sprint review	MS%	EOS review	ES%
Richard Andrášik (Scrum Master)	Uzavrenie zápisníc, metodík a sprintov	Dokumenty už majú rovnaký formát	40%	Všetky dokumenty dopísané a správne formátované	100%
Samuel Kačeriak	Analýza minuloročných scenárov, maličkosti vo frontende	Pridal názvy pre šípky, opravil P2P kontajner, vypoľ bodky	20%	Analýza stále nedokončená, frontend dokončený	75%
Andrej Dubovský	Zlepšuje UX pre finálny prototyp aplikácie	Prototyp je možné zväčšiť a zmenšiť	25%	Zoom funguje dokonale, opravil veci po Ondrovi	100%
Roman Grom	Pospájať všetko na jednotnú dokumentáciu	Zhromaždil potrebné dáta	10%	Dokončil dokumentáciu, docker endpoint navyše	100%
Ondrej Martinka	Opravovanie po ostatných, docker	Docker done, robil si veci do frontendu ako sa mu chcelo	45%	Plieskal navyše features hore dole	100%
Vojtech Fudaly	Zvyšovanie funkcionality frontendu	Upravil nastavenia kontajnerov a CSS	30%	Vynášal kontajnery	85%

Zhrnutie:

- Trvanie sprintu: 30.11 – 12.12

Dokončenie prototypu aplikácie 2. Posledný sprint za prvý semester. Vytvorenie dokumentácie spájaním ostatných súborov.

Začiatok sprintu:

Začíname posledný sprint zimného semestra, preto potrebujeme dokončiť plne funkčný prototyp aplikácie. V tomto sprinte je väčšie množstvo náročnejších úloh, ktoré sú vytvorené s tým, že hneď vieme komu budú pridelené. Podľa prvého pohľadu príde tento sprint najnáročnejší, kvôli

veľkému množstvu úloh, čo je potrebné dokončiť pred skončením semestra. Posledný sprint bude trvať len 13 dní z nejakého nepochopiteľného dôvodu.

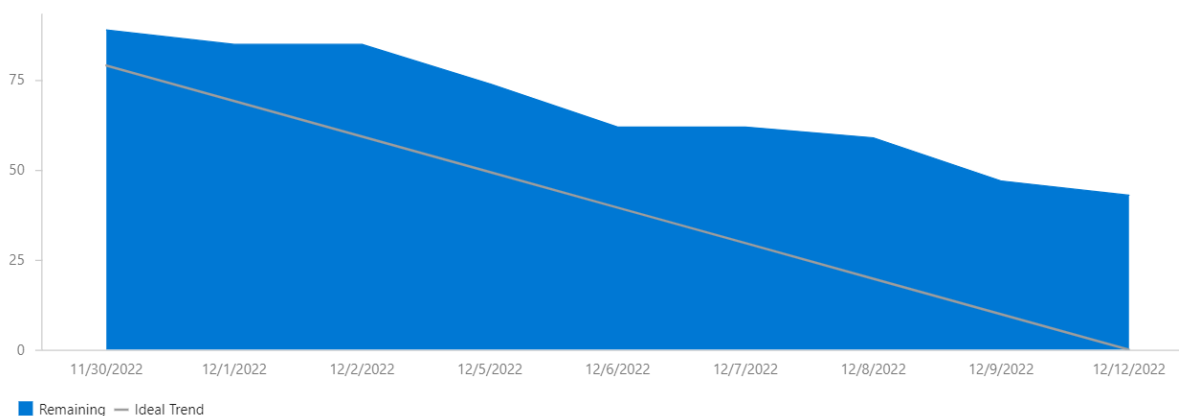
Rekapitulácia v strede sprintu:

V strede tohto sprintu máme len veľmi málo úloh hotových. Semester končí, tak všetci venujeme menej času tímovému projektu a viac času ostatným predmetom. Funkcionalita a výzor aplikácie sa viac menej blíži výsledku za semester, s ktorým by sme boli všetci spokojní. Väčšina úloh síce nie je rozdelená v DevOps ale všetci vieme, ktoré z nich budeme robiť. Posledný týždeň musíme zapnúť, aby sme stihli urobiť všetku tú funkcionality, čo ešte potrebujeme.

Reflexia sprintu:

Tento sprint končil v pondelok, namiesto utorok, čo nás všetkých zaskočilo. Na konci sprintu sme nemali všetky úlohy hotové. To bolo dôsledkom potreby odovzdať väčšieho množstva práce na ostatné predmety a zle odhadnuté, koľko úloh stihneme urobiť pri väčšom množstve práce. Prototyp je dokončený, ale chceli sme urobiť množstvo vedľajšej funkcionality, čo sme žiaľ nestihli. Po tomto sprinte nasledujú medzi semestrami zimné prázdniny, počas ktorých budeme pokračovať na projekte metodikov waterfall. Práca na projekte počas tohto obdobia bude pre všetkých členov tímu nepovinná.

Burndown graf:



Graf vyzerá síce zle, ale urobili sme viac úloh ako vyzerá.

7 ZÁPISNICE

7.1 Zápisnica – stretnutie 1

Dátum:

11.10.2022

Dochádzka:

Andrej Dubovský: Prítomný

Ondrej Martinka: Prítomný

Vojtech Fudaly: Prítomný

Roman Grom: Prítomný

Richard Andrášik: Prítomný

Samuel Kačeriak: Prítomný

Zhrnutie stretnutia:

Zoznámili sme sa s naším vedúcim tímu a diskutovali sme, o tom, čo bude zahŕňať práca na našom tímovom projekte.

Priebeh stretnutia:

- Zoznámenie sa
- Diskusia o skúsenostiach s agile development prístupom
- Všeobecné informácie ohľadom organizácie tímového projektu
- Pridelenie úlohy scrum mastera a zapisovateľa mne
- Rozdelenie úloh do nasledujúceho stretnutia
- Ceremoniálne jedenie koláča

7.2 Zápisnica – stretnutie 2

Dátum:

18.10.2022

Dochádzka:

Andrej Dubovský: Neprítomný

Ondrej Martinka: Prítomný

Vojtech Fudaly: Prítomný

Roman Grom: Prítomný

Richard Andrášik: Prítomný

Samuel Kačeriak: Prítomný

Zhrnutie stretnutia:

Rozbehnutie šprintu Homemasters, približné rozdelenie úloh a zodpovedností pri šprinte. Diskusia o tímovej stránke

Priebeh stretnutia:

- Rekapitulácia práce z minulého týždňa: virtuálne stroje a tímová stránka
- Čas rozbehnúť prvý šprint. Potrebné každému niečo priradiť
- Diskusia o tímových projektoch minulého roku ako materiáloch pre inšpiráciu
- Rozdelenie úloh v šprinte a zvážiť ako dlho potrvá ich vykonanie
- Plán na zdieľanie výsledkov minuloročnej práce na VANET projekte.
- Riešenie rozdelenie úloh a manažment šprintu pomocou Azure DevOps
- Vytvorili sme prvý šprint a pridelili úlohy členom tímu.
- Cieľom tohto šprintu je vytvoriť domovskú stránku a oboznámiť sa s technológiami, s ktorými budeme pracovať
- Každý člen tímu musí pridať výstupy z všetkých úloh na ktorých bude pracovať

7.3 Zápisnica – stretnutie 3

Dátum:

25.10.2022

Dochádzka:

Andrej Dubovský: Neprítomný

Ondrej Martinka: Prítomný

Vojtech Fudaly: Prítomný

Roman Grom: Prítomný

Richard Andrášik: Prítomný

Samuel Kačeriak: Prítomný

Zhrnutie stretnutia:

Stretnutie v strede sprintu, kde sme si uvedomili, že máme už väčšinu úloh hotových. Diskutovali sme všeobecne o tom, ako sa projekt bude vyvíjať.

Priebeh stretnutia:

- Rekapitulácia ako teraz vyzerá webstránka a ukážka nových funkcionalít
- Každý pri stole hovorí, kto čo robil vo sprinte za posledný týždeň
- Chýbajúca analýza minuloročných tímových projektov
- Andrej Dubovský prišiel 9:35, počíta sa stále ako neprítomnosť
- Prešli sme si cez splnené úlohy a celkový progress prvého sprintu
- Diskusia o neschopnosti oficiálnych prednášok Tímového projektu
- Pre náš project je možná aj inšpirácia iného grafického rozhrania pre ns-3, ak existuje
- Potrebné zdefinovať metodiky pre náš tím
- Inšpirácia pre metodiky z projektov minulých rokov

7.4 Zápisnica – stretnutie 4

Dátum:

3.11.2022

Dochádzka:

Andrej Dubovský: Prítomný(online)

Ondrej Martinka: Prítomný

Vojtech Fudaly: Prítomný

Roman Grom: Prítomný

Richard Andrášik: Prítomný

Samuel Kačeriak: Prítomný

Zhrnutie stretnutia:

Reflexia prvého sprintu po jeho dokončení, rozdelenie si úloh na druhý sprint.

Priebeh stretnutia:

- Andrej Dubovský: neprítomný osobne – bol prítomný online
- Každý v tíme povedal spätnú väzbu o prvom sprinte a vlastný prínos
- Členovia tímu vymýšľajú úlohy pre druhý sprint
- Charakterizácia základu projektu
- Vymýšľanie ako bude vyzeráť wireframe a endpointy aplikácie
- Konzultácia o technických podmienkach projektu
- Ustanovenie si, čo chceme dosiahnuť po ďalšom sprinte

7.5 Zápisnica – stretnutie 5

Dátum:

8.11.2022

Dochádzka:

Andrej Dubovský: Prítomný

Ondrej Martinka: Prítomný

Vojtech Fudaly: Prítomný

Roman Grom: Prítomný

Richard Andrášik: Prítomný

Samuel Kačeriak: Prítomný

Zhrnutie stretnutia:

Blížšie rozdelenie úloh pri druhom sprinte. Rozdelenie pozícií v tíme a reflexia v strede sprintu.

Priebeh stretnutia:

- Ondrej ukázal jednoduchý prototyp backendu pre generovanie pcap súborov
- Rozoberanie tímovej stránky, a postup práce na nej
- Diskusia o ďalších úlohách v druhom sprinte
- Potreba vytvorenia základu aplikácie
- Vyberanie frameworku pre frontend aplikácie
- Vojto sa podujme na celý frontend do dokončenia sprintu
- Samo a Andrej sa z dobroty srdca rozhodli pomôcť Vojtovi s frontendom
- Začiatok práce na frontende priamo na stretnutí
- Možnosť rozšírenia funkcionality o akceptovanie výstupu zo SUMO

7.6 Zápisnica – stretnutie 6

Dátum:

15.11.2022

Dochádzka:

Andrej Dubovský: Prítomný

Ondrej Martinka: Prítomný

Vojtech Fudaly: Prítomný

Roman Grom: Prítomný

Richard Andrášik: Prítomný

Samuel Kačeriak: Prítomný

Zhrnutie stretnutia:

Uzatvorenie sprintu 2, otvorenie sprintu 3. Prešli sme si všetko, čo potrebujeme pre minimum na dokončenie predmetu.

Priebeh stretnutia:

- Rekapitulácia funkcionality pridaných hodnôt z posledného sprintu
- Nikto nemá rád Andrejov príspevok v sprinte 2
- Ondrej počul od vedúceho, že bezpečnosť aplikácie nie je kritická
- Každý povedal, čo dosiahol počas posledného sprintu
- Potrebný zoznam všetkého, čo náš simulátor podporuje
- V rámci wireframe, je potrebné vytvoriť aspoň 3 okná
- Vymýšľanie nových úloh, ktoré potrebujeme urobiť ale neboli v sprinte 2
- Sledovanie burndown grafov minulých sprintov
- Diskusia o prioritách úloh a poradí, v akom by sme ich mali urobiť
- Návrh merania úloh v entropiách od Sama Kačeriaka
- Testovanie experimentálnych filtrov na Samovi
- Zhrnutie vybraných a pridelených úloh na koniec

7.7 Zápisnica – stretnutie 7

Dátum:

22.11.2022

Dochádzka:

Andrej Dubovský: Prítomný

Ondrej Martinka: Prítomný

Vojtech Fudaly: Prítomný

Roman Grom: Neprítomný

Richard Andrášik: Prítomný

Samuel Kačeriak: Prítomný

Zhrnutie stretnutia:

Rekapitulujeme sprint 3, Ondrej vysvetlí všetkým, čo majú robiť, spomíname na dobré časy. Nasľubovali sme si funkcionality, čo nemáme šancu do konca semestra stihnúť.

Priebeh stretnutia:

- Roman je neprítomný z dôvodu choroby
- Rekapitulácia, čo kto už urobil v tomto sprinte
- Diskusia o pripravenosti funkčného prototypu aplikácie
- Andrej je v strede sprintu stále zmätený a Ondrej ho musí zachraňovať
- Ondrej zhrnie funkcionality editoru vo frontende
- Vojto ukazuje najhlbšie zákutia frontend kódu
- Robíme tu naživo midsprint code review
- Zmena Github privilégií pre všetkých

7.8 Zápisnica – stretnutie 8

Dátum:

29.11.2022

Dochádzka:

Andrej Dubovský: Prítomný

Ondrej Martinka: Prítomný

Vojtech Fudaly: Prítomný

Roman Grom: Prítomný

Richard Andrášik: Prítomný

Samuel Kačeriak: Prítomný

Zhrnutie stretnutia:

Zrýchlené stretnutie v Coworking miestnosti na FIIT. Rozdeľovanie úloh pre sprint 4 a ukončenie sprintu 3.

Priebeh stretnutia:

- Zmena miestnosti z dôvodu neprítomnosti vedúceho tímu
- Stretnutie urýchlené z dôvodu neprítomnosti vedúceho aby sme mohli ísť skôr domov
- Vymýšľanie nových úloh pre nový sprint
- Návrhy na písanie dokumentácie
- Potrebná analýza minuloročných scenárov a vytvorenie JSON
- Rozdelenie všetkých úloh podľa záujmu a priorít
- Predvedenie prototypu aplikácie vedúcemu tímu
- Vytvorenie prvotných plánov pre ďalší semester

7.9 Zápisnica – stretnutie 9

Dátum:

6.12.2022

Dochádzka:

Andrej Dubovský: Prítomný

Ondrej Martinka: Prítomný

Vojtech Fudaly: Prítomný

Roman Grom: Prítomný

Richard Andrášik: Prítomný

Samuel Kačeriak: Prítomný

Zhrnutie stretnutia:

Preberali sme, ako budeme robiť dokumentáciu, kto spíše frontend a kto backend. Prezentovali sme rozrobené funkcionality prototypu našej aplikácie vedúcemu.

Priebeh stretnutia:

- Mysleli sme si, že sme videli ducha Vojtecha, ale bol živý
- Ukážka nových funkcionalít frontendu

- Ondrej predvádza exportovanie scenáru do python súboru
- Vedúci vymenoval protokoly, ktoré musíme ošetriť v sieti
- Diskusia o spájaní ktorých dokumentov na dokumentáciu
- Sme dohodnutí na poslednom stretnutí tohto semestra; ďalší utorok
- Stretnutie bolo skončené skoro, lebo sme všetci chceli ísť domov

7.10 Zápisnica – stretnutie 10

Dátum:

13.12.2022

Dochádzka:

Andrej Dubovský: Prítomný

Ondrej Martinka: Prítomný

Vojtech Fudaly: Prítomný

Roman Grom: Prítomný

Richard Andrášik: Prítomný

Samuel Kačeriak: Prítomný

Zhrnutie stretnutia:

Posledné stretnutie tohto semestra a ukončenie sprintu 4. Energia je vysoká a hladina nedokončeného burndown grafu ešte vyššia.

Priebeh stretnutia:

- Otvorenie posledného stretnutia za semester
- Potrebne pridať dodatočné veci do dokumentácie, ako návod na spustenie
- Samo má meltdown kvôli git rebase, čo Andrej nechce použiť
- Andrej má nedostatky v jeho zoome
- Naživo zhromažďovanie analýz a update pre dokumentový github
- Diskusia o tom, čo je regulérne vedená Vojtom
- Výber oslavnej reštaurácie