VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ

FACULTY OF ELECTRICAL ENGINEERING AND COMMUNICATION

ÚSTAV TELEKOMUNIKACÍ

DEPARTMENT OF TELECOMMUNICATIONS

ODLIŠNOSTI VERZÍ NÁSTROJE NETWORK SIMULATOR 3 A NOVÁ KONCEPCE VYBRANÝCH LABORATORNÍCH ÚLOH

DIFFERENCES BETWEEN VERSIONS OF NETWORK SIMULATOR 3 AND THE NEW DESIGN OF SELECTED LAB TASKS

BAKALÁŘSKÁ PRÁCE

BACHELOR'S THESIS

AUTOR PRÁCE — Marek Halaš

AUTHOR

VEDOUCÍ PRÁCE — Ing. Jan Dvořák, Ph.D.

SUPERVISOR

BRNO 2024

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ

Bakalářská práce

bakalářský studijní program **Telekomunikační a informační systémy**

Ústav telekomunikací

Student: Marek Halaš ***ID:*** 221274

Ročník: 3 ***Akademický rok:*** 2023/24

NÁZEV TÉMATU:

Odlišnosti verzí nástroje network simulator 3 a nová koncepce vybraných laboratorních úloh

POKYNY PRO VYPRACOVÁNÍ:

Nastudujte z literatury možnosti simulačního nástroje network simulator 3 (ns3) a jeho jednotlivých verzí. Zaměřte se zejména na konkrétní starší verzi (3.21) a pak také nejnovější verzi. Popište hlavní rozdíly v koncepci a odlišnosti v chování těchto verzí. V rámci bakalářské práce přepracujte poskytnuté kódy související s konkrétními laboratorními úlohami předmětu Pokročilé komunikační techniky z verze 3.21 do nejnovější verze dle pokynů vedoucího. Součástí dále bude úprava koncepce dvou daných laboratorních úloh vedoucí vždy k vytvoření předvytvořených scénářů, které se budou v rámci daného návodu určeného pro studenty obohacovat o další související problematiku a využívat další dostupné a relevantní nástroje dle domluvy s vedoucím práce. V praktické části tak vznikne ke každé úloze výchozí zdrojový kód (v nejnovější verzi ns3) spolu s popisem jednotlivých úkonů a nastavením systému ve kterém budou úlohy realizovány. Dále bude praktická část obsahovat podrobný návod v českém jazyce, v rámci kterého budou studenti výchozí kód doplňovat a také samostatné úkoly související s danou problematikou. Rozsah každého z návodů musí být přibližně 2 hodiny času řešení. V rámci ověření funkčnosti a správnosti návodů budou vytvořená řešení vhodně otestována.

DOPORUČENÁ LITERATURA:

[1] Kurose, J. F., Ross, K. W., Computer networking: a top-down approach. 7th global ed. Essex: Pearson, 2017, 852 s. ISBN 978-1-292-15359-9.

[2] JEŘÁBEK, J. Pokročilé komunikační techniky. Skriptum FEKT Vysoké učení technické v Brně, 2020. s. 1-180.

Termín zadání: 5.2.2024 ***Termín odevzdání:*** 28.5.2024

Vedoucí práce: Ing. Jan Dvořák, Ph.D.

prof. Ing. Jiří Mišurec, CSc.
předseda rady studijního programu

ABSTRAKT

Cieľom tejto práce je popísať vývoj simulátora NS3, konkrétne medzi verziami 3.21 a 3.40. Ďalším cieľom bolo vypracovať tri laboratórne úlohy, ktoré boli vyvinuté v staršej verzii a implementovať ich do najnovšej verzie s novými návodmi. V práci sa tiež popisuje nastavenie prostredia a inštalácia tohto nástroja. Posledná časť je zameraná na rozšírenie pôvodných úloh a ich implementáciu do najnovšej verzie.

KĽÚČOVÉ SLOVÁ

NS3, Simulátor počítačových sietí, ISO/OSI, Fragmentáca, IPv4, IPv6

ABSTRACT

The aim of this work is to describe the development of the NS3 simulator, specifically between versions 3.21 and 3.40. Another objective was to prepare three laboratory tasks that were developed in an older version and implement them into the latest version with new guides. The work also describes the environment setup and installation of this tool. The final part focuses on extending the original tasks and their implementation into the latest version.

KEYWORDS

NS3, Computer Network Simulator, ISO/OSI, Fragmentation, IPv4, IPv6

Vysadené pomocou balíčku `thesis` verzie 4.09; `https://latex.fekt.vut.cz/`

HALAŠ, Marek. *Odlišnosti verzí nástroje network simulator 3 a nová koncepce vybraných laboratorních úloh*. Bakalárska práca. Brno: Vysoké učení technické v Brně, Fakulta elektrotechniky a komunikačních technologií, Ústav telekomunikací, 2024. Vedúci práce: Ing. Ján Dvořák, Ph.D.

Vyhlásenie autora o pôvodnosti diela

Meno a priezvisko autora:	Marek Halaš
VUT ID autora:	221274
Typ práce:	Bakalárska práca
Akademický rok:	2023/24
Téma závěrečnej práce:	Odlišnosti verzí nástroje network simulator 3 a nová koncepce vybraných laboratorních úloh

Vyhlasujem, že svoju záverečnú prácu som vypracoval samostatne pod vedením vedúcej/cého záverečnej práce, s využitím odbornej literatúry a ďalších informačných zdrojov, ktoré sú všetky citované v práci a uvedené v zozname literatúry na konci práce.

Ako autor uvedenej záverečnej práce ďalej vyhlasujem, že v súvislosti s vytvorením tejto záverečnej práce som neporušil autorské práva tretích osôb, najmä som nezasiahol nedovoleným spôsobom do cudzích autorských práv osobnostných a/alebo majetkových a som si plne vedomý následkov porušenia ustanovenia § 11 a nasledujúcich autorského zákona Českej republiky č. 121/2000 Sb., o práve autorskom, o právach súvisiacich s právom autorským a o zmene niektorých zákonov (autorský zákon), v znení neskorších predpisov, vrátane možných trestnoprávnych dôsledkov vyplývajúcich z ustanovenia časti druhej, hlavy VI. diel 4 Trestného zákonníka Českej republiky č. 40/2009 Sb.

Brno ....

podpis autora*

*Autor podpisuje iba v tlačenej verzii.

POĎAKOVANIE

Rád by som sa poďakoval vedúcemu bakalárskej práce pánovi Ing. Jánovi Dvořákovi, Ph.D. za odborné vedenie, konzultácie, trpezlivosť a návrhy pre inováciu simulačných úloh.

Obsah

Zoznam obrázkov

Úvod

V súčasnej dobe sú počítačové siete všade okolo nás. Vývoj počítačovích sietí ide stále ku predu a preto sa súčasne musia vyvíjať a vylepšať simulátory na simulovanie týchto sietí. Jedným zo simulátorov počítačových sietí je aj bezplatný nástroj Network Simulátor verzie 3. Tento nástroj umožnuje modelovanie a simuláciu rôznych počítačových scenárov, čím pomáha optimalizovať sieť skôr ako bude reálne implementovaná. Zároveň sa tento nástroj používa pri vzdelávaní, kde je možné bezplatne vytvárať rôzne simulácie, čo vedie k lepšiemu porozumeniu.

Táto bakalárska práca sa zameriava na aktualizáciu simulátora a rozšírenie pôvodných simulačných scenárov využívaných pri vyučovaní. Prvá časť práce je teoretická. Zameriava sa na vysvetlenie počítačových sietí a ich vlastní, ktoré sa používajú v jednotlivých úlohách. Ďalšia časť sa zameriava vývojom tohto simulátora, kde je popísaný ich vývoj a možné vyhliadky do budúcnosti. Súčasťou tejto kapitoli je popis rozdielov medzi verziou 3.21, ktorá bola vydaná roku 2013 a najnovšou verziou 3.40, ktorá bola vydaná koncom roka 2023. Ďalšia kapitola sa zameriava na prípravu operačného systému a samotnú inštaláciu simulátora. Samotný simulátor je pôvodnej verzii obmedzený, kvôli rýchlosti kompilovania. Z tohto dôvodu je potrebné upraviť kompilačné skripty aby bolo možné využívať viaceré nástroje. Okrem samotného simulátora je tu popísaná aj inštalácia nástrojov NetAnim, Wireshark a GNUplot, ktoré sú potrebné pre jednotlivé simulácie. Poslednou časťou sú popísané a implementované nové simulačné scenáre vo vybraných laboratórnych úlohách. Cieľom týchto úloh je, aby študent pochopil správanie a implementovanie simulátora. Zároveň by si týmto študent mal rozšíriť znalosti o počítačových sieťach a taktiež znalosť o pragramovacom jazyku C++.

1 Referenčné modely

Jedná sa o štrukturované schémy, ktoré slúžia pre návrh a implementáciu počítačových sietí. Ich cieľom je poskytnúť spôsob ako popísať a organizovať funkcie siete aby mohli byť aplikované na rôznych platformách s použitím rôznych technológií. Dva najznámejšie sú modely sú ISO/OSI a TCP/IP.

1.1 Model ISO/OSI

ISO/OSI je model sieťovej architektúry, ktorý sa už takmer nepoužíva, pretože bol nahradený efektívnejšími modelmi. Stále ale slúži ako dobrý príklad pri edukácií. Skladá sa zo siedmich vrstiev, kde každá vrstva má svoju úlohu. Jednotlivé vrstvy tohto modelu je vidieť na obr. 1.1.

Obr. 1.1: Vrstvy referenčného modelu ISO/OSI.

1.1.1 Aplikačná vrstva

Najvyššou vrstvou je aplikačná vrstva. Táto vrstva umožnuje používatelovi prístup k počítačovej sieti. Táto vrstva priamo komunikuje s aplikáciami. Môže zabezpečovať napríklad prístup k serveru, posielanie emailov a podobne. Aplikačná vrstva využíva niekoľko protokolov. Medzi najznámejšie patria HTTP, FTP, SMTP, DNS, DHCP. [1]

1.1.2 Prezentačná vrstva

Úlohou tejto vrstvy je zabezpečit a formátovať dáta pre ďalšie vrstvy. Zaisťuje šifrovanie, kompresiu a kódovanie dát. Prijaté dáta z aplikačnej vrstvy sa formátujú tak, aby mali správnu syntax pre nasledujúce vrstvy a koncový bod vedel tieto dáta správne preložiť. Používa protokoly ako napríklad NCP, TLS, SLL. [2]

1.1.3 Relačná vrstva

Relačná vrstva má za úlohu vytvoriť spojenie medzi zariadeniami. Je zodpovedná za udržiavanie tohto spojenia a jeho následné ukončenie. Pre udržanie tohto spojenia využíva kontrolné body. Zabezpečuje komunikáciu vo forme full-duplex alebo half-duplex. Je zodpovedná za riadenie prístupu, napríklad pomocou tokenov. Tiež má na starosti kontrolu autentifikácie a autorizácie. Využíva protokoly ako napríklad RTCP, PPTP, PAP. [3], [4]

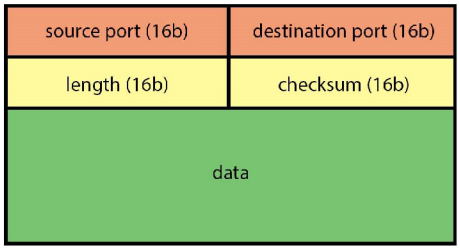
1.1.4 Transportná vrstva

Transportná vrstva je známa ako end-to-end vrstva. Jej úlohou je vytvoriť spojenie medzi zdrojovom a cieľom. Zo správy z vyšších vrstiev rozdelí dáta a vytvorí segmenty. Segment má rôzne podoby na základe použitého protokolu. Poznáme dva základné protokoly transportnej vrstvy, TCP a UDP. [5]

Protokol UDP

Je to bezspojový protokol, ktorý nenadväzuje žiadne spojenie medzi odosielateľom a prijímateľom. Každý UDP segment tu je obslúžený samostatne. Tým, že nie je nadviazané spojenie, tak udp segmenty môžu prísť v inom poradí ako boli odoslané. Taktiež sa môžu tieto segmenty stratiť. UDP segmenty sa považujú za úspešne odoslané hneď ako opustia zariadenie odosielateľa. Výhodou tohto protokolu je rýchlosť. Tým, že sa nenadväzuje žiadne spojenie, tak nevzniká žiadne oneskorenie. Segmenty sa môžu odosielať multicastovo alebo broadcastovo. Ide o realtime prenos. Používa sa napríklad na streamovanie multimédií. Formát UDP segmentu je jednoduchý a môžeme ho vidieť na obr. 1.2.

- Source port - zdrojový port,
- destination port - cieľový port,
- length - dĺžka,
- checksum - kontrólny súčet,
- data - dáta vyšších vrstiev.

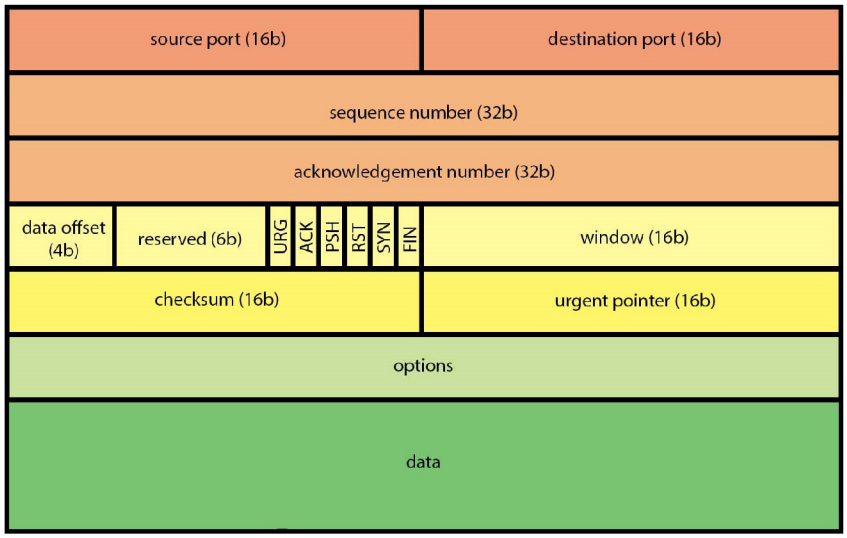


Obr. 1.2: Formát UDP segmentu.

Protokol TCP

Narozdiel od protokolu UDP, tento protokol je spojový. Pred každým posielaním dát je vytvorené spojenie medzi jedným odosielateľom a jedným príjemcom. Takže tento protokol funguje princípom Point to Point a dáta sa tu posielajú v oboch smeroch. Spojenie sa vytvorí pomocou troch správ. Odosielateľ pošle SYN segment. Keď ho príjemca príjme, odošle naspäť SYN, ACK segment. Keď ho odosielateľ príjme odošle ACK segment. Po tomto je vytvorené spojenie a môžu si medzi sebou posielať dáta. Podobným spôsobom sa spojenie aj uzatvára. Pošle sa FIN segment a po odpovedi FIN, ACK segmentu je spojenie ukončené. Použitie protokolu TCP znamená, že ide o potvrdzovaný prenos dát. Segmenty sú označované a teda vždy prídu v správnom poradí. Ak sa nejaký segment stratí, odošle sa znovu. Tento protokol sa používa tam, kde je nutné zaistiť, aby bola správa bezchybná. Môže ísť napríklad o posielanie emailov alebo prácu s databázami. Tým, že je každá správa potvrzdovaná, nie je komunikácia tak rýchla ako u UDP protokolu. TCP počíta aj čas od odoslania segmentu po jeho spätné potvrdenie. Tento čas sa nazýva RTT (round-trip time). Musí sa tu riešiť aj kontrola toku dát alebo kontrola zahltenia siete. Samotný TCP segment je oproti UDP segmentu komplikovanejší. Vzor tohto segmentu je možné vidieť na obr. 1.3. [6]

- Source port - zdrojový port,
- destination port - cieľový port,
- sequence number - sekvenčné číslo,
- acknowledgement number - číslo potvrdenia,
- data offfset - dĺžka hlavičky v bajtoch,
- reserved - rezervované (neobsahuje nič),
- flags - príznaky:
  - urg - urgentné poslanie segmentu,
  - ack - pre potvrdenie,
  - psh - pre okamžité posunutie aplikačnej vrstve,
  - rst - pre tvrdé ukončenie spojenia,
  - syn - pre synchronizáciu,



Obr. 1.3: Formát TCP segmentu.

  - fin - pre označenie paketov na ukončenie spojenia,
- window - veľkosť okna,
- checksum - kontrólny súčet,
- urgetn pointer - poradie bajtu, kde končia urgentné data,
- options - dodatočné nastavenia,
- data - dáta vyšších vrstiev.

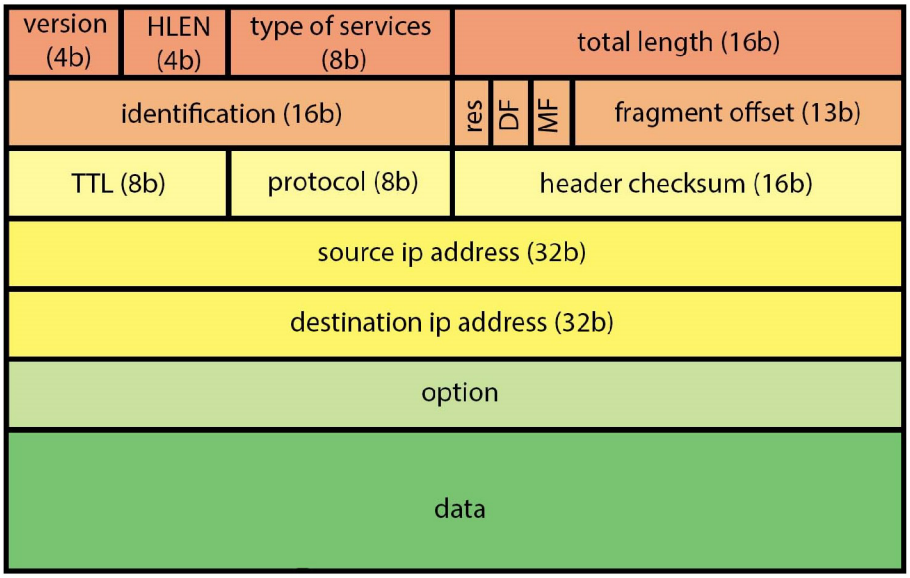
1.1.5 Sieťová vrstva

Sieťová vrstva slúži pre dopravenie datagramu od odosielateľa k prijímatelovi a to čo najefektívnjšie, teda najrýchlejšou cestou. Sieťové zariadenie dostáva velké množstvo datagramov a musí zabezpečiť aby sa každý odoslal správnou cestou. Tieto zariadenia využívajú smerovaciu tabuľku. Smerovacia tabuľka obsahuje rôzne cesty, kadiaľ môžu byť datagramy odoslané. Každá cesta má ale inú hodnotu (metriku). Kedže sa sieť neustále mení, tak je potrebné aby boli tieto smerovacie tabulky stále aktualizované. Obsah smerovacích tabuliek upravujú smerovacie algoritmy, ktoré sú pravidelne vysielané po sieti, aby reagovali na zmeny v sieti, výpadky, vytvorenia spojení atď.. Príkladom smerovacieho algoritmu môže byť protkol ICMP. Najčastejšie sa používa na správy o nedostupnosti siete. Hlavička tohto protokolu sa odlišuje na základe konkrétneho typu správy, avšak jej veľkosť je vždy 64 bitov.

V tejto vrstve sa pridá datagramu ďalšia hlavička. Hlavička má rôzne podoby na základe použitého protokolu IP (Internet Protocol). [7]

IPv4

Hlavička tohto protokolu obsahuje pomerne veľa informácií. Je dlhá 20 až 40 B a používa 32 bitové adresy. Dĺžka tejto adresy obmedzuje maximálne množstvo 4,3 miliardy možných adries. Hlavičku tohto protokolu je vidieť na obr. 1.4.



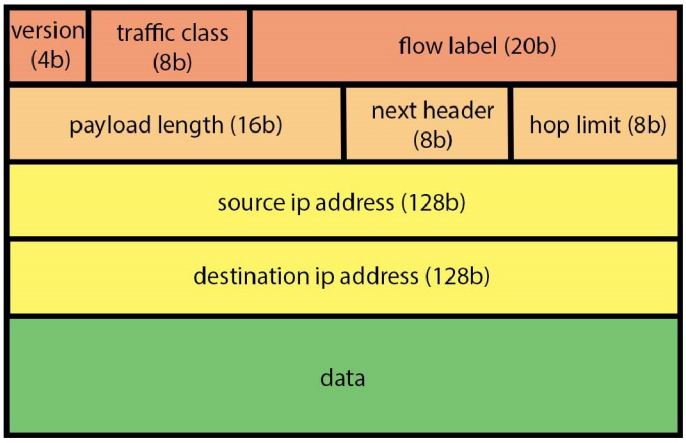
Obr. 1.4: Formát IPv4 datagramu.

- Version - verzia protokolu IP,
- HLEN - dĺžka hlavičky,
- type of services - určenie priority,
- total length - dĺžka celého paketu,
- identificator - identifikácia pri fragmentovaní,
- flags - príznaky o fragmentácií:
  - res - rezervované,
  - DF - označuje že nemôže byť fragmentovaný,
  - MF - označuje či sú ďalšie fragmenty za týmto paketom,
- fragment offset - pozícia fragmentu v pakete,
- TTL - počet možných skokov,
- protocol - typ protokolu v segmente,
- header checksum - kontrolný súčet,
- source ip address - zdrojová ip adresa,
- destination ip address - cielová ip adresa,
- options - dodatočné nastavenia,
- data - dáta vyšších vrstiev.

IPv6

Tým, že IPv4 je obmedzená množstvom IP adries, tak bola vytvorená nová verzia tohto protokolu. Datagram tohto protokolu má dĺžku 40 B. Táto verzia používa 128 bitové adresy. Týmto je zabezpečené "nekonečné" množstvo adries. Ako môžeme vidieť na obr. 1.5, tak z pôvodnej hlavičky IPv4 bolo odstránených pomerne veľa častí a je teda aj jednoduchšia.

- Version - verzia použitého protokolu,
- traffic class - určenie priority,
- flow label - identifikuje tok dát,
- payload length - veľkosť datagramu v bajtoch,
- next header - označenie typu nasledujúcej hlavičky,
- hop limit - počet možných skokov,
- source ip address - zdrojovú ip adresu,
- destionation ip address - cieľovú ip adresu,
- data - dáta vyšších vrstiev.



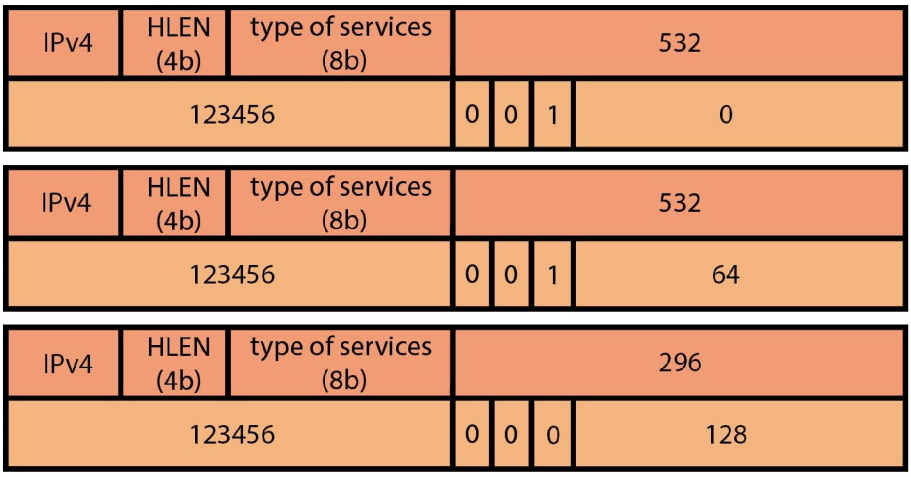
Obr. 1.5: Formát IPv6 datagramu.

Fragmentácia pri IPv4

Každý smerovač je obmedzený maximálnou veľkosťou paketu (MTU - Maximum Transmission Unit), ktorú môže odoslať do ďalšej siete. Ak je paket príliš veľký, môže ho rozdeliť na množinu menších fragmentov a poslať ich postupne. Fragmetny postupujú ako autonómne pakety a konečný príjemca je zodpovedný za opätovné zostavenie týchto fragmentov späť do pôvodného paketu. Fragmentácia je v datagrame zabezpečená 32-bitovým poľom, ktoré je rozdelené na 3 podpolia. Prvé pole je 16-bitový identifikátor paketu. Zabezpečuje aby všetky fragmenty boli označené ako fragmenty konkrétneho pôvodného paketu. Druhým poľom sú 3-bitové príznaky.

Prvý bit sa nepoužíva. Druhým bitom je DF (dont fragment), označuje či sa môže daný paket fragmentovať a ak nie tak sa tento paket zahodí. Tretí bit, MF (more fragment) označuje, že za týmto fragmentom nasleduje ešte ďalší fragment. Len posledný fragment z paketu má tento bit nastavený na 0 a označuje, že je posledný. Tretie pole označuje pozíciu fragmentu v pôvodnom pakete. Označuje sa pomocou oktetov. [8]

Príklad fragmentácie v IPv4: Je paket o dĺžke 1300 bajtov. Smerovač v sieti ale zvláde len pakety o dĺžke 532 bajtov. Takže paket sa rozdelí na tri fragmenty. Prvý fragment bude mať dĺžku 532 bajtov. Kedže je to prvý fragment, tak hodnota fragment offset bude mať hodnotu 0. Hodnota MF je nastavená na 1, pretože ešte nasledujú ďalšie fragmenty. Ďalší fragment bude mať tiež 532 bajtov. Predchádzajúci fragment mal dĺžku 532 bajtov ale 20 bajtov bola hlavička. Takže tento fragment začína od 512 bajtu, čo predstavuje hodnotu fragment offset 64. Za týmto ešte bude nasledovať ďalší fragment takže hodnota MF je stále nastavená na 1. Týmto sa prenieslo 1024 bajtov z pôvodného paketu. Posledný fragment bude mať dĺžku už len 296 bajtov (20 bajtov pre hlavičku datagramu). Fragment offset bude mať hodnotu 128 a kedže to je posledný fragment tak hodnota MF bude 0. Znázornenie jednotlivých fragmentov je vidieť na obr. 1.6.



Obr. 1.6: Fragmenty pôvodného paketu, pri použití IPv4.

Nevýhodou fragmentácie môže byť napríklad pri používanom protokole TCP. Ak sa nejaký fragment stratí, musia sa znovu poslať úplne všetky fragmenty. Toto pomerne dosť zvyšuje záťaž na sieti. Preto sa fragmentácia u TCP zvyčajne nepoužíva pretože si dohodnú veľkosť paketov pri vytvorení spojenia. Ďalšou nevýhodou môže byť, že fragmenty sa môžu použiť pri DDOS útokoch.

Fragmentácia pri IPv6

Fragmentácia v IPv6 je komplikovanejšia. Pôvodný zámer bol aby fragmentácia vôbec nenastávala. Všetky pakety majú označení príznak DF na 1, teda nefragmentovať. Fragmentácia je teda možná len na koncových zariadeniach. Avšak pôvodný zámer nevyšiel a fragmentáciu je nutné občas vykonať aj pri IPv6. Ak teda príde paket, ktorý je príliš veľký, odošle sa naspäť ICMP paket s kódom Packet Too Big (príliš velký paket). Zariadenie ktoré paket vytvorilo ho môže prefragmentovať na menšie časti. Nevýhodou fragmentácie v IPv6 je filtrovanie ICMP paketov. Ak sa tento paket zahodí nezistí sa že velký paket bol zahodený. Toto spôsobuje problém napríklad pri použití protokolu UDP. Paket sa považuje za úspešene odoslaní v momente keď opustí router. Po odoslaní je vymazaní z vyrovnávacej pamäte a teda ICMP PTB už nepomôže. [8]

Aby sa predišlo réžii a nemusela sa fragmentácia meniť na každom uzli, tak sa používa technika Path MTU Discovery. Na začiatku sa odošle veľký paket. Ak sa paket nestratí, znamená to, že veľkosť paketu je správna. Ak sa paket stratí, znamená to, že niekde na ceste je uzol, ktorý nezvládne takúto veľkosť paketu. Keď sa paket stratí, automaticky sa u odosielateľa zmenší veľkosť paketu. Toto sa opakuje až kým sa nenájde vhodná veľkosť paketu. [9]

1.1.6 Spojová vrstva

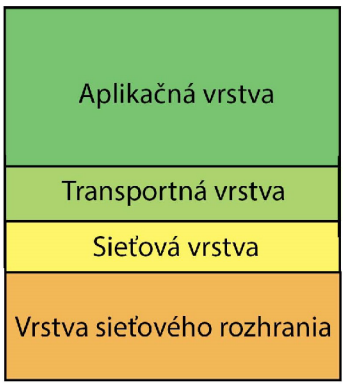
Úlohou spojovej vrstvy je zabezpečiť bezchybný prenos dát. Môže to zabezečiť rôznimi spôsobmi. Napríklad pomocou multiplexingu alebo pomocou nastavenia prenosovej rýchlosti. Kedže sa môžu počas prenosu dáta poškodiť vplyvom šumu alebo útlmu, je potrebné tieto dáta zakódovať s použitím samoopravých kódov. Ďalej zabezpečuje aby nevznikali kolízie. Ak je viacero zariadení pripojených k rovnakému komunikačnému kanálu tak môžu vznikať. Je to zabezpečené prístupovými metódami ako napríklad CSMA/CA (Carrier Sense Multiple Access / Collision Avoidance) K datagramu sa tu pridáva hlavička s MAC adresou, ktorá je nutná pri riadení ramcov. [10]

1.1.7 Fyzická vrstva

Je to najnižšia vrstva. Nezaoberá sa dátami aké má. Jej jediným cielom je aby preniesla dáta cez fyzické médium. Aby sa mohli jednotlivé bity prenášať, je potrebné ich dostať do signálu. K tomu sa používajú na zmenu amplitúdy, frekvencie a fázy nosného signálu. Používajú sa aj ich rôzne kombinácie pre dosiahnutie najlepšej možnej modulácie, napríklad QAM.

1.2 TCP/IP

Tento model vychádza z pôvodného modelu ISO/OSI. Teda zabezpečuje všetko čo bolo spomínané v sekcii 1.1. Zatiaľ čo model ISO/OSI mal presne špecifikované, čo sa na ktorej vrstve deje, tu sú vrstvy viac flexibilné. To znamená, že niektoré vrstvy môžu používať funkcie z iných vrstiev. Tým sa stal viac univerzálnejším a rýchlejším. Obsahuje len štyri vrstvy. Niektoré vrstvy boli zachované, iné boli zlúčené. Aplikačná, prezentačná a relačná boli zlúčené do aplikačnej vrstvy. Spojová a fyzická boli zlúčené do vrstvy sieťového rozhrania. Zvyšné vrstvy zostali zachovalé. Aktuálne to je najpoužívanejší referenční model. Tento model je znázornený na obr. 1.7.



Obr. 1.7: Vrstvy referenčného modelu TCP/IP.

2 Počítačové siete

Sieť a komunikácia v nej môže mať veľa tvarov. Môžeme ju rozdeliť podľa velkosti. Kde niektoré sú určené pre malé oblasti alebo naopak pre velké, ktoré spájajú celé kontinenty. Taktiež každá sieť može mať inú topológiu. Tá opisuje štruktúru a spôsoby komunikácie. Zariadenia môžu byť rovnocenné ale može tu byť aj hierarchické. Zariadenia v sieti môžu komunikovať s kýmkoľvek. Podľa potreby môžu zariadenia komunikovať vzájomne alebo so všetkými naraz.

2.1 Typy sieťových pripojení

V rámci rozsahu a využitia môžeme sieť rozdeliť do štyroch základných kategórií. Znázornenie týchto kategórií je vidieť na obr. 2.1.

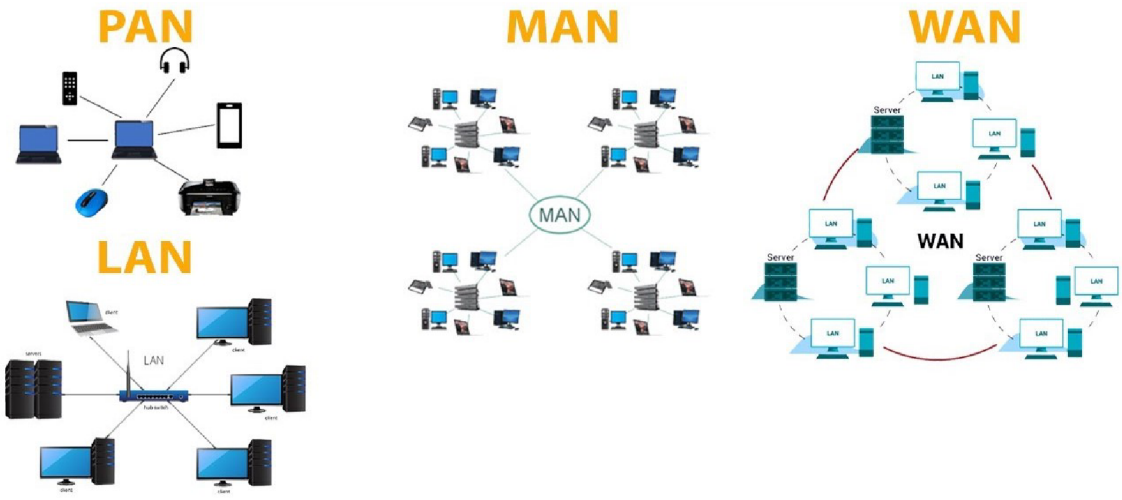
WAN (Wide Area Network) Jedná sa o typ siete ktorá je najväčšia. Umožnuje prepojenie medzi menšími sieťami ako sú MAN alebo LAN. Využívajú rôzne prenosové média. Prenosová rýchlosť oproti sieťam MAN a LAN je menšia, dôvodom je vzdialenosť, čím sa zväčšuje latencia. Príkladom tejto siete môžu byť medzištátne firmy, ktoré spolu vzájomne komunikujú. Táto sieť tiež zabezpečuje vzdialený prístup alebo cloudový prístup.

MAN (Metropolitan Area Network) Táto sieť je menšia ako sieť WAN a väčšia ako sieť LAN. Zvyčajne pokrýva oblasti do niekoľkých desiatok kilometrov ako sú napríklad mestá alebo metropolitné oblasti. Slúži na prepájanie sietí LAN, čím sa zvyšuje a zefektívňuje komunikácia. Táto sieť umožňuje prenos na väčších rýchlostiach a to do niekolko Gbps. Typickým príkladom tejto siete sú telefónny operátory, ktorý poskytujú internetové pripojenie. [11]

LAN (Local Area Network) Táto sieť pokrýva malú oblasť. Používajú sa najmä v domácnostiach, kanceláriach alebo podobných oblastiach. Prenosová rýchlosť v tejto sieti môže dosahovať niekolko Gbps. Kedže je táto sieť malá a zariadenia sú pomerne blízko sebe, latencia v týchto sieťach je veľmi nízka. Používa sa napríklad na prepojenie počítačov, kameier, tlačiarní alebo iných sieťových zariadení. Základom každej LAN siete je smerovač. Ten smeruje komunikáciu medzi zariadeniami v sieti alebo sa prepája s ďalšími sieťami. Oproti iným sieťam je táto sieť viac odolná voči chybám. Táto sieť môže fungovať aj bezdrátovo, teda nie je potrebné žiadne fyzické médium. Takáto sieť sa označuje ako WLAN. [12]

PAN (Personal Area Network) Jedná sa o najmenšiu sieť. Často pokrýva len oblasť niekolkých metrov. Je určená na prepojenie malého množstva zariadení ako sú mobilné telefóny, tablety, počítače a podobne. Umožňuje komunikáciu priamo medzi zariadeniami bez potreby prístupového bodu. Používa

sa prevažne bezdrátová komunikácia. Nevýhodou tejto siete môže byť rušenie z okolitých sietí. Najčastejšie sa vyskytuje pri technológii bluetooth.



Obr. 2.1: Typy počítačových sietí. Obrázok prevzatý z [17].

2.2 Spôsovy distribúcie dát

Jedná sa o spôsoby akými sa dáta v sieťach doručujú na základe IP adresovania a smerovania. Medzi základné štyri typy patria: unicast, broadcast, multicast a anycast.

Unicast Ide o priamu komunikáciu medzi jedným odosielateľom a jedným príjemcom. Každý paket má presne definovanú zdrojovú a cieľovú IP adresu. Takže pre každý komunikujúci pár je nutné zaistiť svoju vlastnú komunikáciu. Používa sa najmä pre súkromné komunikácie ako napríklad posielanie emailov, pozeranie filmu alebo synchronizovanie jednotlivých uzlov siete.

Broadcast Pri broadcaste odosielateľ odosiela pakety na všetky zariadenia v sieti. Posielajú sa bez ohľadu na to, či sú dáta určené pre jednotlivé uzly. Na odoslanie sa používajú špeciálne broadcastové adresy. Pre IPv4 siete sú dva typy broadcastových adries. Jeden typ pre obmedzený broadcast. Táto adresa je vždy 255.255.255.255 a označuje že sa majú dáta odoslať všetkých zariadeniam v lokálnej sieti. Druhým typom je sieťový broadcast. Sieťový broadcast odosieľa všetkým zariadeniam v celej sieti. Adresa pre tento broadcast môže byť rôzna. Na adresu siete sa aplikuje maska siete a tým získame broadcostovú adresu. Napríklad v sieti 192.168.88.0/24 bude broadcastová adresa 192.168.88.255. V rámci IPv6 sieti sa broadcast nepoužíva. Kvôli bezpečnosti bol nahradený

multicastom. Broadcast sa používa napríklad na informovanie všetkých zariadení o chybách v sieti alebo o dostupnosti. Často sa používa pri ARP protokole na získanie MAC adries.

Multicast Multicast umožňuje odosielať pakety len vybranej skupine zriadení, ktoré sa zaregistrovali, aby boli súčasťou danej skupiny. Na odoslanie sa používajú multicastové adresy. V rámci IPv4 sú multicastové adresy v rozsahu od 224.0.0.0 po 239.255.255.255. Multicastové adresy sa v rámci IPv4 rozdeľujú na tri základné skupiny. Adresy v rozsahu od 224.0.0.0 do 224.0.0.255 slúžia len pre lokálne sieťové protokoly a slúžia teda len na komunikáciu v rámci lokálnej sieti. Adresy v rozsahu od 224.0.1.0 do 238.255.255.255 sa používajú na smerovanie vrámci celého internetu. Posledné adresy sú v rozsahu od 239.0.0.0 do 239.255.255.255.. Tieto adresy sú určené len pre interné sieťové aplikácie. Vrámci IPv6 majú adresy istú štruktúru. Vždy začínajú prefixom FF. Nasledujú 4 bity, ktoré určujú či sú adresy trvalé alebo dočasné. Potom nasledujú ďalšie 4 bity, ktoré určujú rozsah v ktorom sú adresy platné. Po tomto už nasleduje len 112 bitové identifikačné číslo skupiny. Rozsah FF02::/16 slúži len v rámci lokálnej siete. Pre komunikáciu vrámci jednej siete sa používa rozsah FF05::/16. A pre komunikáciu v internetu sa používa rozsah FF0E::/16. [13] [14]

Anycast Používa sa na posielanie jednému zo skupiny cieľových uzlov. Všetky zariadenia v skupine majú rovnakú IP adresu. Keď sa dáta majú odoslať, odošlú sa na najbližší uzol zo skupiny. Najbližší uzol sa určuje na základe metriky. Vďaka tomuto sa vie zabezpečiť väčšia dostupnosť a výkonnosť siete. Anycast sa používa napríklad pri cloudových službách alebo DNS serveroch.

3 Network Simulator (NS)

Network Simulátor (ďalej NS) je nástroj ktorý slúži na simulovanie počítačovej siete. Ide o nástroj, ktorý je bezplatný a je 'open-source'. Vďaka tomuto nástroju môže uživateľ navrhnúť počítačovú sieť v rôznych prostrediach ako sú aj bezdrátové siete. Samotný nástroj nie je viazaný na žiadny hardware. Je určený najmä pre výskum a vzdelávanie. Samotný nástroj prešiel veľkým vývojom.

3.1 História a vývoj nástroja NS3

Tento nástroj prešiel mnohými verziami. Dali by sa rozdeliť na 4 základné verzie, ktoré sú na sebe neviazané, teda boli vyvíjané neviazane na predchádzajúce verzie. Najnovšia používaná verzia je aktuálne NS3, ktorá bola vydaná v roku 2008.

NS1 Jedná sa o prvú verziu tohto simulačného nástroja. Táto verzia bola implementovaná len v rámci programovacieho jazyka TLC. Tento nástroj automaticky prepočítaval trasu a nepodporoval smerovacie protokoly. Táto verzia je však už len minulosťou a boli vytvorené modernejšie.

NS2 Tento nástroj už bol vyvinutý pomocou programovacieho jazyka C++. Podporuje simuláciu v reálnom čase. Umožnuje už použitie protokolov ako TCP, UDP, FTP a podobné. Oproti nástroju NS1 už umožnuje aj grafické zobrazenie simulovanej siete. Ponúka dve grafické zobrazenia. Net, ktorý zobrazuje topologický návrh siete a samotný priebeh simulácie. Xgraph slúži na zobrazenie grafu, ktorá predstavuje výsledky vo forme grafu. Tento nástroj slúži prevažne pre menšie siete, pri väčších simuláciach je tento nástroj nespoľahlivý. Tento nástroj už prestáva byť podporovaný a nemá dostatočnú dokumentáciu.

NS3 Táto verzia už používa jazyk C++. Umožnuje používať API pre používanie skriptov v pythone. Oproti NS2 je tento nástroj lahšie používateľný a má oveľa lepšiu dokumentáciu. Tento nástroj dokáže zvládnuť aj zložitejšie simulácie. Používa Netanim pre zobrazenie simulácie. Ďalej umožnuje zaznamenať celú simuláciu napríklad do programu wireshark, kde môže byť následne lepšie analyzovaný. Aktuálne je najdostupňejšia verzia 3.41. Samotná práca je však vykonaná s použitím verzie 3.40, ktorá bola vydaná v septembri roku 2023.

NS4 Táto verzia je zatiaľ vo vývoji. Používa pomerne nový jazyk P4 ktorý je priamo orientovaný na siete. Táto verzia by mala mať možnosť importovať konkrétne hardwarové zariadenia aby bola simulácia jednoduchšia a menej náročná na implementáciu. Cieľom tejto verzie je, aby bol simulátor pre uživateľa príjemnejší a prechod zo simulácie do reálneho použitia čo najlahší.[15]

3.2 Hlavné rozdiele medzi verziami 3.21-3.40

Základné funkcie, ktoré boli definované vo verzii 3.21 boli takmer nezmenené. Prebehli tu len drobné rozšírenia knižníc alebo pridanie presnejších definícií vlastností siete ako napríklad odstárnenie limitu MTU pre UDP paket, posielanie nekonečeného množstva paketov a podobné drobné úpravy. Ďalej umožňuje pridávať tagy pre priority paketov. [16]

Zlepšilo sa správanie TCP protoklu. TCP protokol už dokáže kontrolovať preťaženie siete. Dokáže upravovať prenosovú rýchlosť na základe veľkosti okna. Ďalej TCP už sám ukončuje spojenie po niekoľkých neprijatých paketoch. Celkovo je teda vyvinuté lepšie riadenie TCP komunikácie. [16]

Od tejto verzie sa vývoj zameral na nové rozšírenia. Bolo vytvorených mnoho funkcií pre simulovanie televíznych vysielačov. Ďalej tu bola implementovaná nová funkcionalita Nix-Vector smerovania, ktorá zjednodušuje smerovanie tým, že sa snaží predpovedať optimálnu cestu. Ďalšia časť rozšírenia bola zameraná na simulovanie bezdrátovej komunikácie, kde bolo vytvorených niekoľko knižníc, ktoré slúžia na lepšie simulácie a ich správanie. [16]

Okrem rozširovania možností simulácie bolo niekoľko zmien aj v zostavení a správaní samotného simulátora. Najnovšia verzia podporuje štandard C++ verzie 17. Namiesto Python2 podporuje už aj Python3. Bolo upravených niekoľko vlastností tak, aby namiesto chybnej simulácie nebolo možné simuláciu ani skompilovať. Tým sa zabezpečilo ľahšie debugovanie. K lepšiemu debugovaniu bolo upravené aj hľadanie prepojení závislostí jednotlivých knižníc. S debugovaním pomáha aj nová funkcionalita trasovania zahodených paketov. Zjednodušila sa tu práca s dokumentami .csv a bolo vytvorených niekoľko nových vzorových simulácií. Najväčšou zmenou bola kompilácia simulácie. Do verzie 35 bol upravovaný .waf skript pre efektívnejšie kompilovanie. Od verzie 36 sa zmenil spôsob kompilovania a .waf skript bol nahradený nástrojom Cmake, ktorý používa odlišné konfiguračné súbory. [16]

4 Praktická časť

4.1 Prostredie pre NS3 a inštalácia potrebných komponentov

Všetky úlohy a samotná inštalácia simulátora je realizovaná vo virtuálnom operačnom systéme Linux s verziou 20.04.2. Pre účely laboratórnych úloh je potrebné nainštalovať okrem samotného simulátora aj nástroje ako je NetAnim, Gnuplot alebo Wireshark

4.1.1 Inštalácia NS3.40

Celá inštalácia je vykonaná v príkazovom riadku. Najskôr sa zadá príkaz, ktorý stiahne najnovšie informácie o dostupných balíčkoch a verziach.

```
$ sudo apt update
```

Následne nainštalujeme balíčky ktoré sú nevyhnutné pre simulátor ako je napríklad kompilátor pre jazyk C++, nástroj na zostavenie software a podobne. Inštalujú sa týmto príkazom.

```
$ sudo apt install g++ python3 cmake ninja-build git gir1.2-
   goocanvas-2.0 python3-gi python3-gi-cairo python3-
   pygraphviz gir1.2-gtk-3.0 ipython3 tcpdump wireshark
   sqlite sqlite3 libsqlite3-dev openmpi-bin openmpi-common
   openmpi-doc libopenmpi-dev doxygen graphviz imagemagick
   python3-sphinx dia imagemagick texlive dvipng latexmk
   texlive-extra-utils texlive-latex-extra texlive-font-utils
    libeigen3-dev gsl-bin libgsl-dev libgslcblas0 libxml2
   libxml2-dev libgtk-3-dev lxc-utils lxc-templates vtun uml-
   utilities ebtables bridge-utils libxml2 libxml2-dev
   libboost-all-dev ccache
```

Keď sú všetky baličky nainštalované, môže sa nainštalovať samotný simulátor. V príkazovom riadku je potrebné presunúť sa na pracovnú plochu. Tu sa stiahne samotný simulátor. Simulátor sa stiahne ako zazipovaný, preto je potrebné ho aj odzipovať.

```
$ cd Desktop
$ wget https://www.nsnam.org/releases/ns-allinone-3.40.tar.
   bz2
$ tar xfj ns-allinone-3.40.tar.bz2
```

Pred nasledujúcim krokom je potrebné upraviť konfiguračný súbor tak aby bolo možné používať nástroj NetAnim. V zložke ns-3.40 je potrebné otvoriť textový dokument s názvom CMakeLists.txt. V tomto dokumente je potrebné upraviť riadok kde je nastavenie pre NetAnim, ktoré sa nachádzan a 53. riadku. Tento riadok nám povolí používanie tohto nástroja. Inštalácia tohto nástroja je popísaná v sekcii 4.1.2.

```
option(NS3_NETANIM "Build netanim" ON)
```

Po tomto je nutné presunúť sa do zložky simulátora, kde sa zadá príkaz, ktorý pripravi konfiguráciu zdrojových súborov pred ich samotnou kompiláciou.

```
$ cd ns-allinone-3.40/ns-3.40
$ ./ns3 configure --enable-examples --enable-tests
```

Po tomto by mal byť simulátor nainštalovaný. Je možné to overiť kompilovaním predpripravených simulácií a ich následným spustením. Do verzie 3.35 sa pre kompiláciu používal skript waf. Od tejto verzie sa na kompilovanie používa skript build (prvé kompilovanie môže zabrať desiatky minút).

```
$ ./ns3 build
```

Keď je kompilácia dokončená, tak sa po presune do zložky kde sú vytvorené simulácie môže overiť ich funkčnosť.

```
$ cd build/examples/tutorial
$ ./ns3-40-first-default
```

4.1.2 Inštalácia NetAnim

NetAnim je nástroj, ktorý slúži na vizualizáciu simulácie sieťových topológií a prevádzky v sieti. V tomto nástroji je možné presne usporiadať uzly, sledovať pohyb paketov v danom časovom okamžiku.

Pre inštaláciu tohto nástroja je potrebné nainštalovať ešte niektoré balíky. Opäť sa stiahnu najnovšie novšie informácie o balíkoch a ich verziách, a následne sa nainštalujú potrebné balíky.

```
$ sudo apt update
$ sudo apt install qtbase5-dev qtchooser qt5-qmake qtbase5-
   dev-tools
```

Následne po presune do zložky netanim-3.109 sa použije príkaz, ktorý slúži na konfiguráciu NetAnim pred jeho samotou komipláciou a po tomto sa spustí kompilácia. Týmto je nástroj nainštalovaný a môže sa spustiť.

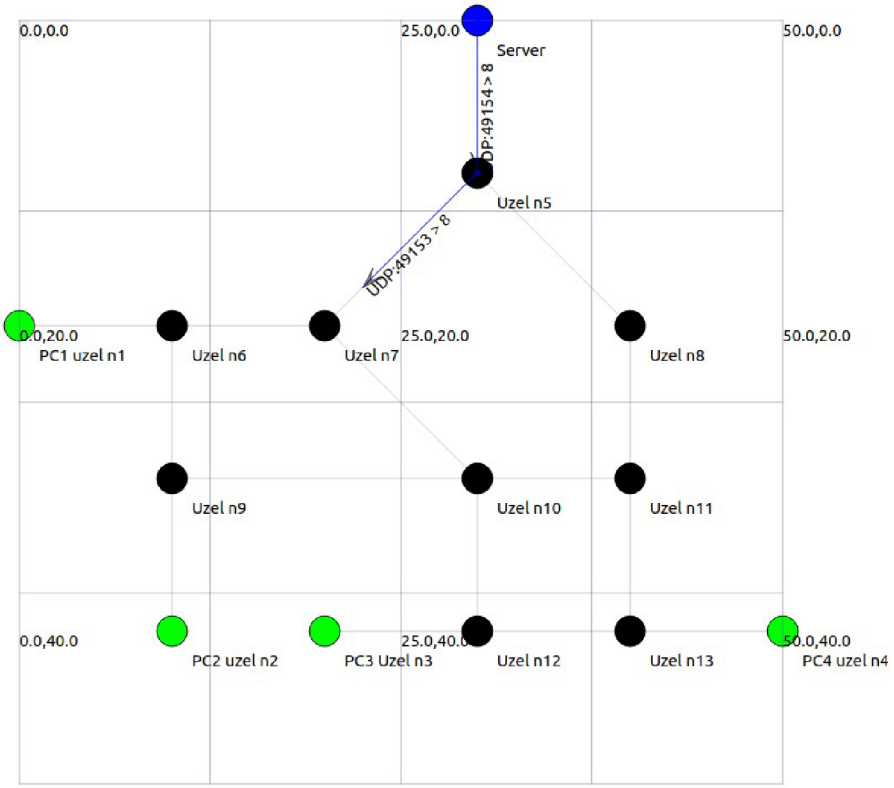
```
$ qmake NetAnim.pro
$ make
$ ./NetAnim
```

Ak je simulácia siete vykonaná v zložke scratch, tak silmulácia bude kompilovateľná. Ak sa použije NetAnim na iných miestach simulátora NS3 tak je potrebné upraviť ďalšie konfiguračné súbory.

Napríklad je potrebné spustiť NetAnim pre predpripravené simulácie, tak je potrebné rozšíriť konfiguračný súbor o prepojenie knižníc. Je to realizované pridaním riadku `$libnetanim`. Celá časť konfiguračného súboru vyzerá nasledovne.

```
build_example(
  NAME first
  SOURCE_FILES first.cc
  LIBRARIES_TO_LINK
    ${libcore}
    ${libpoint-to-point}
    ${libinternet}
    ${libapplications}
    ${libnetanim}
)
```

Výstup tohto nástroja je zobrazený na obr. 4.1.



Obr. 4.1: Grafický výstup nástroja NetAnim.

4.1.3 Inštalácia Gnuplot

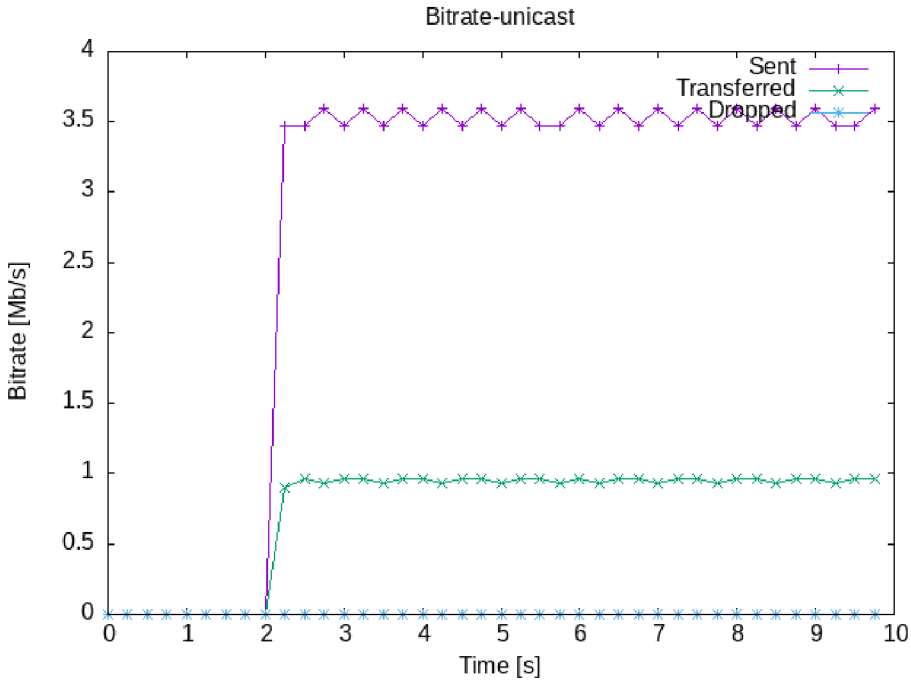
Gnuplot je program, pomocou ktorého je možné vykreslovať 2D a 3D grafiku. Umožňuje vykresľovať grafy napríklad aj pomocou skriptovania. Pre jeho inštaláciu je nutné stiahnuť ďalšie balíky a nainštalovať samotný program.

```
sudo apt update
sudo apt install gnuplot
```

Po tomto je program nainštalovaný. Ak je správne nakonfigurovaná simulácia tak sa po jej spustení vytvorí súbor "name.plt". K vykresleniu do .png formátu sa môže použiť nasledujúci príkaz, ktorý sa použije v zložke, kde je simulácia vykonaná.

```
gnuplot -p name.plt && eog name.png
```

Príkad grafického výstupu je vidieť na obr. 4.2.



Obr. 4.2: Grafický výstup nástroja Gnuplot.

4.2 Inštalácia Wireshark

Wireshark je bezplatný nástroj na analýzu sieťových protokolov. V tomto nástroji je možné zachytiť celú komunikáciu na sieti a analyzovať jednotlivé pakety. Vďaka tomuto je možné debugovať možné chyby vyskytnuté v sieti. Inštalácia tohto nástroja je jednoduchšia. V príkazovom riadku sa opäť zadá príkaz na aktualizáciu informáciách o dostupných balíčkoch a verziách.

```
$ sudo apt-get update
```

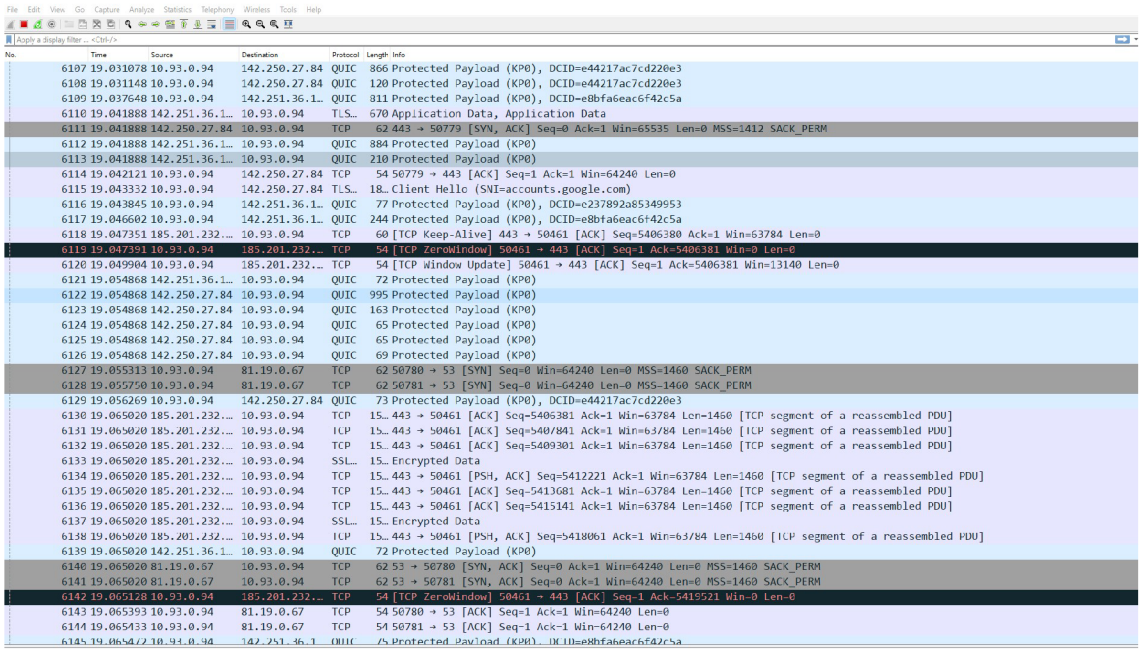
Následne sa príkazom môže spustiť samotná inštalácia.

```
$ sudo apt install wireshark
```

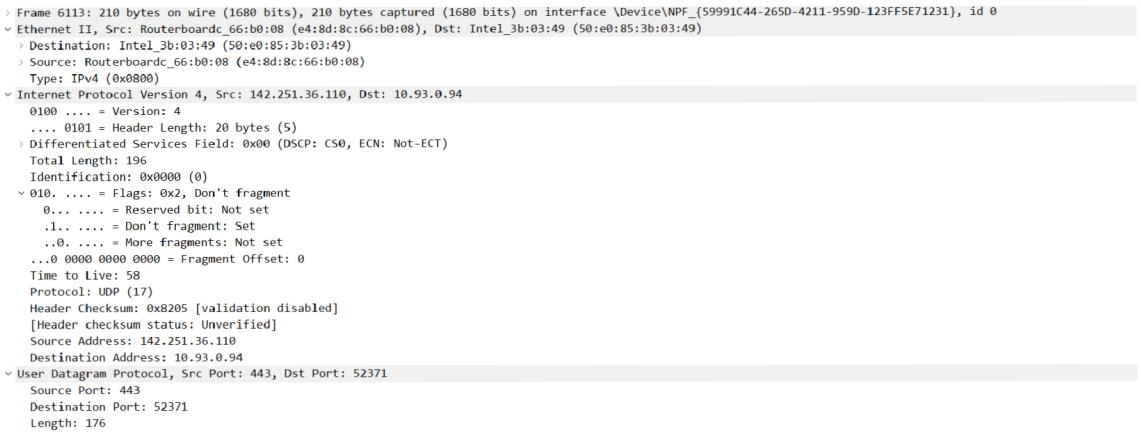
Po dokončení je možné overeiť že wireshark bol správne nainštalovaný. V príkazovom riadku sa zadá príkaz `wireshark`, ktorým sa spustí.

```
$ wireshark
```

Grafické zobrazenie má dve základné časti. Jedna časť zobrazuje všetky zachytené pakety a druhá zobrazuje detail jednotlivých paketov. Grafické zobrazenie je vidieť na obr. 4.3 a 4.4.



Obr. 4.3: Zachytená komunikácia nástrojom Wireshark.



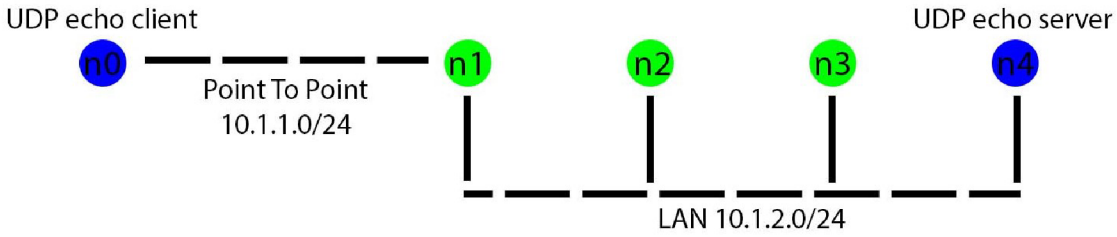
Obr. 4.4: Detail paketu zachytného nástrojom Wireshark.

5 Úloha č.1 - Point to Point

Táto úloha sa zaoberá smerovaním Point to Point. Táto úloha je pomerne jednoduchá a rýchlo dokončiteľná. V úlohe sa nachádza jeden uzol, ktorý komunikuje so zmernicou a s jej ďalšími uzlami. V úlohe sa mení množstvo odoslaných paketov a sleduje sa zväčšujúca hodnota odozvy RTT. Bola pridaná doplňujúca úloha, ktorej cielom bude rozšíriť zbernicu a komunikovať so všetkými uzlami súčasne. Táto doplňujúca úloha nezaberie až tak veľa času, kedže sa bude len upravovať súčasná úloha.

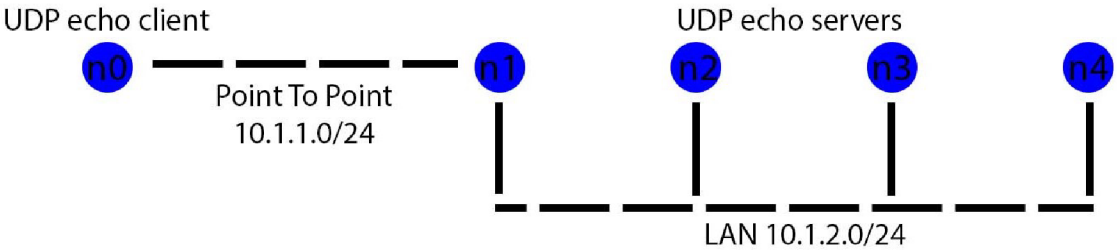
5.1 Zmeny v danej úlohe

V pôvodnej úlohe sa použil model siete, kde sú dva uzly (n0 a n1) prepojené pomocou technológie point to point. Uzol n1 a zvyšné uzly n2, n3 a n4 používajú pre komunikáciu technológiu Ethernet s prístupovou metódou CSMA/CD. Ďalej sa tu vytvoril dátový tok UDP ktorý prebiehal medzi uzlamy n0 a n4. Uzol n4 bol umiestnený v sieti LAN. Topológiu tejto úlohy je vidieť na obr. 5.1.



Obr. 5.1: Pôvodná topológia siete.

Novou úlohou tohto cvičenia je upraviť topológiu tak, aby komunikovali všetky uzli súčasne. Nová topológia teda bude vyzerať ako na obr. 5.2.



Obr. 5.2: Upravená topológia siete.

Úprava sa vykoná na dvoch miestach. Najskôr sa upraví aplikácia vytvorenia echo serverov. Namiesto nastavenia na konkrétny uzol sa vytvorí cyklus, ktorý bude prechádzať cez všetky uzly, kde vytvorí server. Kedže nie je až tak velká záťaž v sieti, zvyšok nie je nutné meniť a aplikácia sa môže vypnúť v čase 10 sekúnd.

```
for (uint32_t i = 0; i <= nCsma; ++i) {
    UdpEchoServerHelper echoServer(7);
    ApplicationContainer serverApps = echoServer.Install(csmaNodes.
        Get(i));
    serverApps.Start(Seconds(1.0));
    serverApps.Stop(Seconds(10.0));
}
```

Ďalej je potrebné vytvoriť echo klientov na každý uzol. Toto bude tiež aplikované pomocou cyklu ktorý prejde všetky uzly a na každom definuje prevoz. Tu tiež nie je nutné meniť zvyšné parametre komunikácie.

```
for (uint32_t i = 0; i <= nCsma; ++i) {
    UdpEchoClientHelper echoClient(csmaInterfaces.GetAddress(i), 7)
        ;
    echoClient.SetAttribute("MaxPackets", UintegerValue(1));
    echoClient.SetAttribute("Interval", TimeValue(Seconds(1.0)));
    echoClient.SetAttribute("PacketSize", UintegerValue(1500));
    ApplicationContainer clientApps = echoClient.Install(p2pNodes.
        Get(0));
    clientApps.Start(Seconds(2.0));
    clientApps.Stop(Seconds(10.0));
}
```

Výsledok tejto rozšírenej simulácie je vidieť na obr. 5.3. Z obrázku je vidieť že všetky pakety boli odoslané v čase 2 sekúnd. Echo servery ich však prijímali postupne.

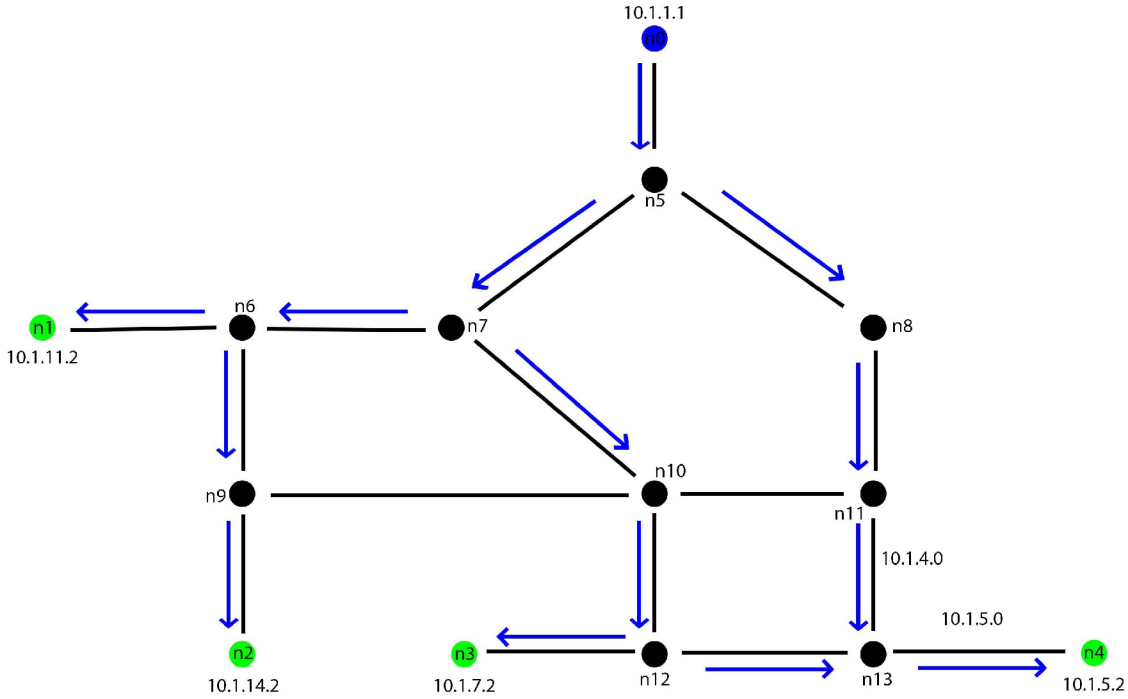
```
student@student-virtual-machine:~/Desktop/ns-allinone-3.40/ns-3.40/build/scratch$ ./ns3.40-vutlogin-default
At time +2s client sent 1024 bytes to 10.1.2.1 port 7
At time +2s client sent 1024 bytes to 10.1.2.2 port 7
At time +2s client sent 1024 bytes to 10.1.2.3 port 7
At time +2s client sent 1024 bytes to 10.1.2.4 port 7
At time +2.00369s server received 1024 bytes from 10.1.1.1 port 49153
At time +2.00369s server sent 1024 bytes to 10.1.1.1 port 49153
At time +2.00737s client received 1024 bytes from 10.1.1.2 port 7
At time +2.01435s server received 1024 bytes from 10.1.1.1 port 49154
At time +2.01435s server sent 1024 bytes to 10.1.1.1 port 49154
At time +2.01738s server received 1024 bytes from 10.1.1.1 port 49155
At time +2.01738s server sent 1024 bytes to 10.1.1.1 port 49155
At time +2.01836s server received 1024 bytes from 10.1.1.1 port 49156
At time +2.01836s server sent 1024 bytes to 10.1.1.1 port 49156
At time +2.02002s client received 1024 bytes from 10.1.2.2 port 7
At time +2.02604s client received 1024 bytes from 10.1.2.3 port 7
At time +2.02903s client received 1024 bytes from 10.1.2.4 port 7
```

Obr. 5.3: Výstup rozšírenej simulácie.

Tu bola úloha ešte rozšírená o samostatnú časť, kde sa bude meniť počet uzlov s ktorými sa komunikuje súčasne. Cieľom tohto je pozorovať ako sa mení hodnota odozvy (RTT) vplyvom zvýšenej záťaže na sieti. Zo simulácie sa následne zistí že táto hodnota narastá konštantne.

6 Úloha č. 2 - Unicast a multicast

Táto úloha simuluje prenos paketov pomocou unicastu a multicastu. Pôvodná úloha obsahovala tri scenáre. Najskôr študenti simulujú unicast a multicast. Pri týchto simuláciách upravujú prenosovú rýchlosť a sledujú zmeny pomocou programu Gnuplot. Tretí scenár simuloval pridanie druhého serveru. Všetky scenáre boli simulované aj pomocou programu NetAnim, kde študenti vidia smerovanie paketov počas simulácie. Topológia a smerovanie pre túto úlohu je zobrazené na obr. 6.1.



Obr. 6.1: Topológia a smerovanie v multicastovom prevoze s centrálnym bodom.

6.1 Zmeny v danej úlohe

Pôvodné zadanie a topológia tejto úlohy bola nezmenená. Bolo tu upravené nastavenie pre frontu paketov, pretože atribút z pôvodnej verzie "MaxPackets" , sa už nepoužíva. Bol nahradený atribútom "MaxSize". Ďalej v definícii StringValue musí byť definované, že sa jedná o pakety. Nové nastavenie vyzerá následovne.

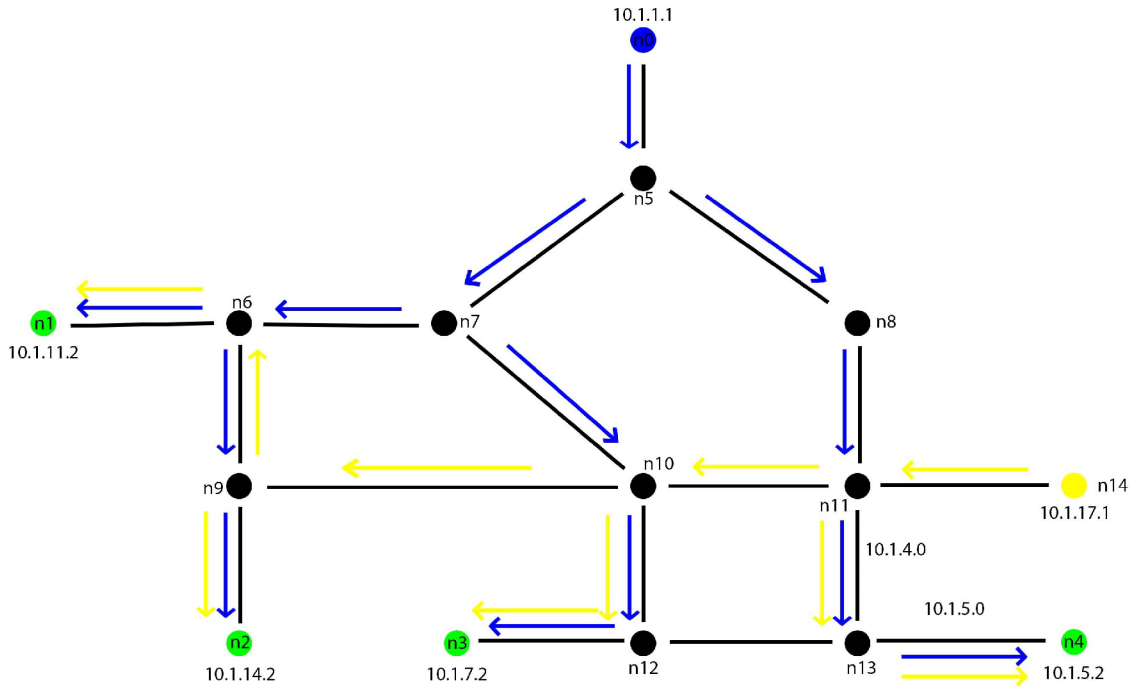
Okrem týchto hlavných zmien boli ešte upravené uzly na niektorých rozhraniach, ktoré mohli byť pre študentov mätúce. Jednalo sa o zmeny v poradí priradenia uzlov, pretože neboli totožné s názvom rozhraní.

```
string queueLength = "10p";
pointToPoint.SetQueue ("ns3::DropTailQueue", "MaxSize", StringValue
    (queueLength));
```

6.1.1 Úprava scenára s dvomi servermi

V tejto úlohe sa používala dopredu vytvorená simulácia. Táto simulácia obsahovala druhý server, ktorý bol vedľa pôvodného serveru. Skupiny zariadení a smerovanie bolo nezmenené.

Zmenou v tejto úlohe je, že sa presunul server na iné miesto ako je znázornené na obr. 6.2.



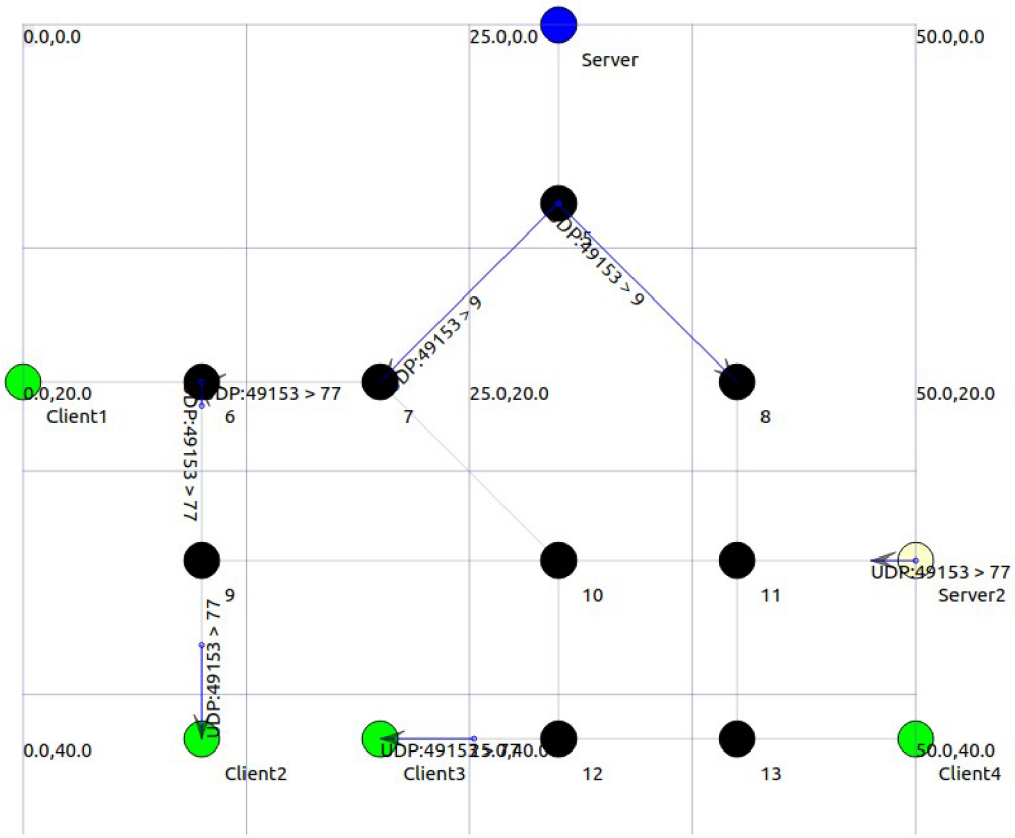
Obr. 6.2: Nová topológia a smerovanie v multicastovom prevoze s dvomi servermi.

Táto zmena vyžadovala vytvorenie nových skupín zariadení a nové smerovanie. Tieto zmeny boli dosiahnuté nasledujúcim nastavením.

```
NetDeviceContainer outputDevices_n11;
outputDevices_n11.Add(netDevices11_13.Get(0));
outputDevices_n11.Add(netDevices11_10.Get(0));
NetDeviceContainer outputDevices_n10;
outputDevices_n10.Add(netDevices12_10.Get(1));
outputDevices_n10.Add(netDevices10_9.Get(0));
NetDeviceContainer outputDevices_n9;
outputDevices_n9.Add(netDevices9_2.Get(0));
outputDevices_n9.Add(netDevices6_9.Get(1));
```

```
multicastRouting2.AddMulticastRoute(nodes.Get(11), multicastSource2
    , multicastGroup2, netDevices14_11.Get(1), outputDevices_n11);
multicastRouting2.AddMulticastRoute(nodes.Get(13), multicastSource2
    , multicastGroup2, netDevices11_13.Get(1), netDevices13_4.Get(0)
    );
multicastRouting2.AddMulticastRoute(nodes.Get(10), multicastSource2
    , multicastGroup2, netDevices11_10.Get(1), outputDevices_n10);
multicastRouting2.AddMulticastRoute(nodes.Get(12), multicastSource2
    , multicastGroup2, netDevices12_10.Get(0), netDevices12_3.Get(0)
    );
multicastRouting2.AddMulticastRoute(nodes.Get(9), multicastSource2,
     multicastGroup2, netDevices10_9.Get(1), outputDevices_n9);
multicastRouting2.AddMulticastRoute(nodes.Get(6), multicastSource2,
     multicastGroup2, netDevices6_9.Get(0), netDevices6_1.Get(0));
```

Táto úloha zostala pre študentov už vypracovaná. Je to z toho dôvodu, že vypracovanie tejto úlohy by zaberalo viac ako 2 hodiny. Vysvelenie implementácie tejto úlohy bolo pridané do návodu. Simuláciu zobrazenú v NetAnim je vidieť na obr. 6.3.



Obr. 6.3: Výstup z programu NetAnim, pri použitý dovch serverov.

6.1.2 Nový scenár s vytvorením centrálneho uzla (rendezvous point

Táto úloha bola rozšírená o úlohy kde sa v multicastovej simulácii vytvorí centrálny uzol. K tomu je potrebné vytvoriť nové skupiny zariadení, ktoré zefektivňujú komunikáciu.

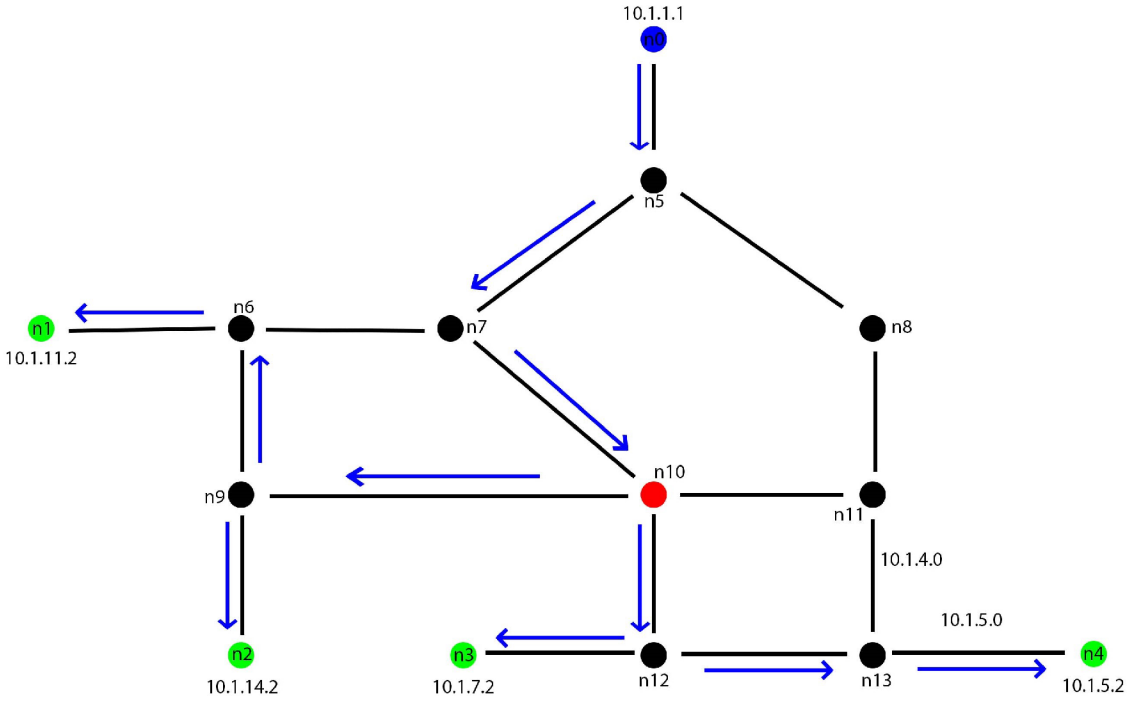
```
NetDeviceContainer outputDevices_n10;
outputDevices_n10.Add(netDevices12_10.Get(1));
outputDevices_n10.Add(netDevices10_9.Get(0));

NetDeviceContainer outputDevices_n12;
outputDevices_n12.Add(netDevices12_3.Get(0));
outputDevices_n12.Add(netDevices13_12.Get(1));
```

```
NetDeviceContainer outputDevices_n9;
outputDevices_n9.Add(netDevices9_2.Get(0));
outputDevices_n9.Add(netDevices6_9.Get(1));
```

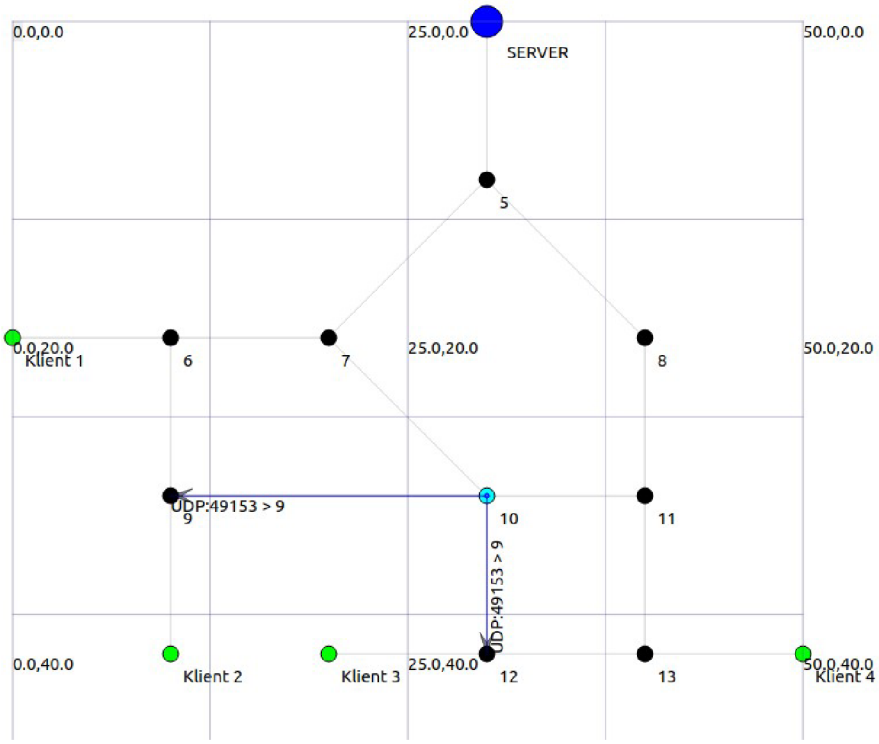
Následne boli upravené smerovania podľa nového scenára. V novom smerovaní teda nie je potrebné smerovať komunikáciu na uzly n11 a n8.

```
NetDeviceContainer outputDevices_n10;
outputDevices_n10.Add(netDevices12_10.Get(1));
outputDevices_n10.Add(netDevices10_9.Get(0));

NetDeviceContainer outputDevices_n12;
outputDevices_n12.Add(netDevices12_3.Get(0));
outputDevices_n12.Add(netDevices13_12.Get(1));

NetDeviceContainer outputDevices_n9;
outputDevices_n9.Add(netDevices9_2.Get(0));
outputDevices_n9.Add(netDevices6_9.Get(1));
```

Znázornenie tejto simulácie je zobrazené na obr. 6.4 a výstup z programu NetAnim je vidieť na obr. 6.5



Obr. 6.4: Topológia a smerovanie v multicastovom prevoze s centránym bodom.



Obr. 6.5: Výstup simulácie v programe NetAnim s použitím centrálneho bodu.

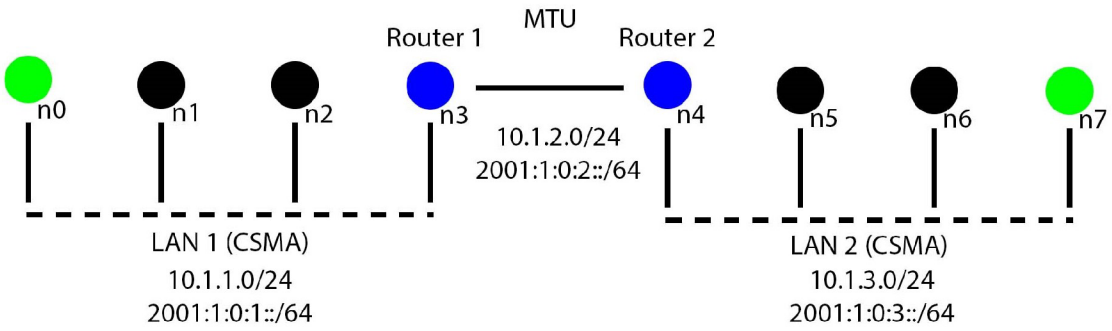
7 Úloha Fragmentácia IPv4 a IPv6

Táto úloha sa zameriava na simulovanie fragmentácie pri použití IPv4 a IPv6. V priebehu úlohy sa mení hodnota MTU na jednotlivých smerovačoch tak, aby bolo dosiahnuté viacnásobného fragmentovania, a teda bola lepšie vysvetlená daná problematika.

Táto úloha sa zameriava na simulovanie siete s použitím IPv4 a IPv6, kde sa mení hodnota MTU a dosahujú sa tým rôzne fragmentácie.

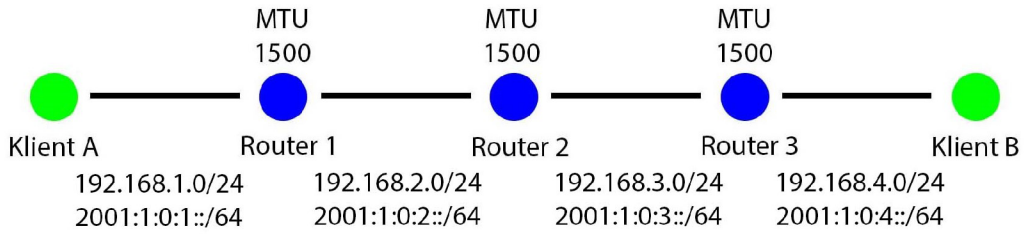
7.1 Zmeny v tejto laboratórnej úlohe

Bol vytvorení nový koncept tejto úlohy. V pôvodnom zadaní boli dve siete LAN ktoré komunikovali pomocou 2 smerovačov, ktoré boli prepojené technológiou bod-bod. Pôvodná topológia tejto úlohy je zobrazená na obr. 7.1.



Obr. 7.1: Pôvodná topológia siete.

V novej úlohe sa nachádzajú dvaja klienti, medzi ktorými sú tri smerovače. Všetky uzly sú prepojené technológiou bod-bod. Nová topológia tejto úlohy je vidieť na obr. 7.2.



Obr. 7.2: Nová topológia siete.

7.1.1 Implementácia novej topológie

Na začiatku boli definované knižnice, povolenie logovania a vytvorenie hlavnej funkcie `main`. Na začiatku bolo potrebné vytvoriť požadované uzly. Tie boli vytvorené pomocou nasledujúceho kódu.

```
Ptr<Node> node_A = CreateObject<Node>();
Ptr<Node> router1 = CreateObject<Node>();
Ptr<Node> router2 = CreateObject<Node>();
Ptr<Node> router3 = CreateObject<Node>();
 Ptr<Node> node_B = CreateObject<Node>();
```

Z týchto uzlov boli následne vytvorené skupiny uzlov aby im bolo možné prideliť adresné rozsahy.

```
NodeContainer A_R1 = NodeContainer(node_A, router1);
NodeContainer R1_R2 = NodeContainer(router1, router2);
NodeContainer R2_R3 = NodeContainer(router2, router3);
NodeContainer R3_B = NodeContainer(router3, node_B);
NodeContainer netAll = NodeContainer(node_A, router1, router2,
    router3, node_B);
```

Následne bola pridaná inštalácia internetového stacku na všetky uzly.

Po tomto sa definovali parametre pre bod-bod komunikáciu, ktorá bola následne nainštalovaná na vytvorené skupiny uzlov. nie je tu definovaná hodnota MTU priamo pre bod-bod komunikáciu ale pre jednotlivé skupiny uzlov samostatne.

```
PointToPointHelper p2p;
p2p.SetDeviceAttribute("DataRate", StringValue("10Mbps"));
p2p.SetChannelAttribute("Delay", StringValue("2ms"));

NetDeviceContainer net_A_R1 = p2p.Install(A_R1);
NetDeviceContainer net_R1_R2 = p2p.Install(R1_R2);
NetDeviceContainer net_R2_R3 = p2p.Install(R2_R3);
NetDeviceContainer net_R3_B = p2p.Install(R3_B);

net_A_R1.Get(1)->SetMtu(1500);
net_R1_R2.Get(0)->SetMtu(1500);
net_R1_R2.Get(1)->SetMtu(1500);
net_R2_R3.Get(0)->SetMtu(1500);
net_R2_R3.Get(1)->SetMtu(1500);
net_R3_B.Get(0)->SetMtu(1500);
```

K jednotlivým skupinám uzlov boli pridelené adresné rozsahy ako pre IPv4 tak pre IPv6.

```
Ipv4AddressHelper ipv4;
ipv4.SetBase("192.168.1.0", "255.255.255.0");
Ipv4InterfaceContainer ipv4_A_R1 = ipv4.Assign(net_A_R1);

ipv4.SetBase("192.168.2.0", "255.255.255.0");
Ipv4InterfaceContainer ipv4_R1_R2 = ipv4.Assign(net_R1_R2);

ipv4.SetBase("192.168.3.0", "255.255.255.0");
Ipv4InterfaceContainer ipv4_R2_R3 = ipv4.Assign(net_R2_R3);

ipv4.SetBase("192.168.4.0", "255.255.255.0");
Ipv4InterfaceContainer ipv4_R3_B = ipv4.Assign(net_R3_B);
```

```
Ipv6AddressHelper ipv6;
ipv6.SetBase(Ipv6Address ("2001:1:0:1::"), Ipv6Prefix("ffff:ffff:
    ffff:ffff::"));
Ipv6InterfaceContainer ipv6_A_R1 = ipv6.Assign(net_A_R1);
ipv6_A_R1.SetForwarding(1, true);
ipv6_A_R1.SetDefaultRouteInAllNodes(1);

ipv6.SetBase(Ipv6Address("2001:1:0:2::"), Ipv6Prefix("ffff:ffff:
    ffff:ffff::"));
Ipv6InterfaceContainer ipv6_R1_R2 = ipv6.Assign(net_R1_R2);
ipv6_R1_R2.SetForwarding(0, true);
ipv6_R1_R2.SetForwarding(1, true);

ipv6.SetBase(Ipv6Address("2001:1:0:3::"), Ipv6Prefix("ffff:ffff:
    ffff:ffff::"));
Ipv6InterfaceContainer ipv6_R2_R3 = ipv6.Assign(net_R2_R3);
ipv6_R2_R3.SetForwarding(0, true);
ipv6_R2_R3.SetForwarding(1, true);

ipv6.SetBase(Ipv6Address("2001:1:0:4::"), Ipv6Prefix("ffff:ffff:
    ffff:ffff::"));
Ipv6InterfaceContainer ipv6_R3_B = ipv6.Assign(net_R3_B);
ipv6_R3_B.SetForwarding(0, true);
ipv6_R3_B.SetDefaultRouteInAllNodes(1);
```

Následne bolo pridané smerovanie. V IPv4 sa definuje smerovanie jedným príkazom. Pri použití IPv6 je potrebné donastaviť smerovanie ručne.

```
Ipv4GlobalRoutingHelper::PopulateRoutingTables();

Ipv6StaticRoutingHelper routingHelper;
Ptr<Ipv6StaticRouting> routingTable = routingHelper.
    GetStaticRouting(router1->GetObject<Ipv6>());
routingTable->AddNetworkRouteTo(Ipv6Address("2001:1:0:4::"),
    Ipv6Prefix(64),ipv6_R1_R2.GetInterfaceIndex(0));
routingTable->AddNetworkRouteTo(Ipv6Address("2001:1:0:1::"),
    Ipv6Prefix(64),ipv6_A_R1.GetInterfaceIndex(1));
routingTable->AddNetworkRouteTo(Ipv6Address("2001:1:0:1::"),
    Ipv6Prefix(64),ipv6_A_R1.GetInterfaceIndex(0));
routingTable = routingHelper.GetStaticRouting(router2->GetObject<
    Ipv6>());
routingTable->AddNetworkRouteTo(Ipv6Address("2001:1:0:1::"),
    Ipv6Prefix(64),ipv6_R1_R2.GetInterfaceIndex(0));
 routingTable->AddNetworkRouteTo(Ipv6Address("2001:1:0:4::"),
     Ipv6Prefix(64),ipv6_R2_R3.GetInterfaceIndex(0));
routingTable = routingHelper.GetStaticRouting(router3->GetObject<
    Ipv6>());
routingTable->AddNetworkRouteTo(Ipv6Address("2001:1:0:1::"),
    Ipv6Prefix(64),ipv6_R1_R2.GetInterfaceIndex(1));
```

Poslednou časťou bolo vytvorenie dvoch aplikácii. Jednu pre IPv4 a jednu pre IPv6.

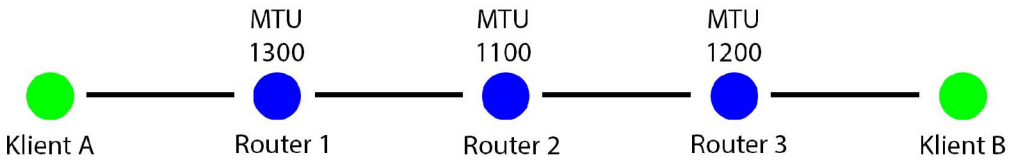
```
UdpEchoServerHelper server(7);
ApplicationContainer serverApps = server.Install(node_B);

UdpEchoClientHelper echoClientIPv4(ipv4_R3_B.GetAddress(1), 7);
echoClientIPv4.SetAttribute("MaxPackets", UintegerValue(1));
echoClientIPv4.SetAttribute("Interval", TimeValue(Seconds(1.0)));
echoClientIPv4.SetAttribute("PacketSize", UintegerValue(5000));
ApplicationContainer clientAppsIPv4 = echoClientIPv4.Install(node_A
    );

UdpEchoClientHelper echoClientIPv6(ipv6_R3_B.GetAddress(0, 1), 7);
echoClientIPv6.SetAttribute("MaxPackets", UintegerValue(1));
echoClientIPv6.SetAttribute("Interval", TimeValue(Seconds(1.0)));
echoClientIPv6.SetAttribute("PacketSize", UintegerValue(5000));
ApplicationContainer clientAppsIPv6 = echoClientIPv6.Install(node_A
    );
```

Následne sa nastavil čas, kedy sa majú jednotlivé aplikácie vykonať. Ďalej bolo pridané zazanamenávanie komunikácie do .pcap súboru na všetky uzly. Je to z toho dôvodu, že fragmentácia sa bude vykonávať na rôznych uzloch a z prvého rozhrania by nebolo možné sledovať postupné fragmentovanie. Po tomto nasledovalo už samotné zapnutie a vypnutie simulátora.

```
serverApps.Start(Seconds(2.0));
serverApps.Stop(Seconds(16.0));
clientAppsIPv4.Start(Seconds(2.0));
clientAppsIPv4.Stop(Seconds(8.0));
clientAppsIPv6.Start(Seconds(8.0));
clientAppsIPv6.Stop(Seconds(16.0));

p2p.EnablePcapAll("vutlogin");
```

7.1.2 Zmeny velkosti MTU na trase

V pôvodnom zadaní boli úlohy ktoré merali hodnotu RTT pred fragmentáciou. Tieto úlohy nesúviseli s fragmentáciou a boli nahradené novými, ktoré sú viac orientované na samotné fragmentovanie. Napríklad pomocou zmeny MTU na jednotlivých smerovačoch ako je znázornené na obr. 7.3.



Obr. 7.3: Topológia siete so zmeneními hodnotami MTU na jednotlivých smerovačoch.

Táto zmena hodnôt zabezpečí mnohonásobnú fragmentáciu, ktorú budú študenti analyzovať. Pri použití IPv4 sa najskôr prefragmentuje na 1300. Na ďalšom smerovači sa tieto fragmenty rozdelia na ďalšie fragmenty podľa nastavenej hodnoty 1100. Táto analýza je sledovaná na každom smerovači zvlášť.

7.1.3 Zmena velkosti MTU na trase pri použití IPv6

Fragmentovanie s použitím IPv6 je odlišné. Fragmentovať sa môže len na koncových zariadeniach a nie na smerovačoch. Takže po prijatí velkého fragmentu sa odošle naspäť správa s príznakom Packet Too Big. Avšak ani aktuálna verzia simulátora nedokáže spracovať tieto správy a nedokáže simulovať ani použitie paketu PMTUD. Takže fragmentovanie IPv6 je dovolené len pri nastavení rovnakej hodnoty MTU po celej trase. Počas vytvárania tejto úlohy som sa pokúsil vytvoriť vlastné knižnice na vytvorenie unikátnych klientov tak, aby dokázali správu s príznakom Packet Too Big spracovať. Toto ale neviedlo k úspešnému cielu. Aj napriek tomuto úloha ponúka dostatočné simulácie na pochopenia fragmentovania.

Závěr

Samotný nástroj prešiel od verzie 3.21 do verzie 3.40 niekoľkými zmenami. Všetky pôvodné funkcie boli takmer nezmenené. Zmenilo sa len niekoľko názvov atribútov a podobne. Okrem drobných zmien bolo vylepšené riadenie pri TCP komunikácii. Bolo pridaných niekoľko nových funkcií, ktoré boli zamerané na bezdrátovú sieť alebo simulovanie predpovedania ideálnej cesty. Velkou zmenou bolo kedy sa simulácie spúštali pomocou skriptu `.waf`. Tento skript bol vyvíjaný aby bol efektívnejší. Neskôr sa však nahradil nástrojom `Cmake`. Všetky simulácie jednotlivých úloh boli vypracované vo virtuálnom operačnom systéme Linux verzie 22.04.2. Predvolený operačný systém nemal všetky potrebné komponenty pre nástroj NS3. Okrem samotnej inštalácie simulátora bolo potrebné nainštalovať potrebné balíky a ďalšie komponenty ako NetAnim, Gnuplot alebo wireshark. Samotný nástroj NS3 bol v predvolenom stave obmedzený. Bolo nutné upraviť konfiguračné súbory aby dokázal pracovať so všetkými funkciami vytvorených simulácií.

V rámci tejto práce boli upravené pôvodné laboratórne úlohy. V prvej úlohe bolo simulovanie novej situácie, kedy všetky zariadenia komunikujú súčasne. V druhej úlohe bola upravená časť simulácie, kde vznikol centálny bod, cez ktorý smerovala celá multicastová komunikácia. Týmto boli vysvetlené nastavenia pri smerovaní. Poslednou časťou úlohy bol stav, kde bol pridaný druhý server. Tento server bol umiestnený na iné miesto v topológii tak, aby využíval vlastné smerovanie. V tretej úlohe sa zmenila topológia siete. V novej topológii sa už nenachádzajú siete LAN. V tejto topológii sú dvaja klienti, ktorý sú prepojený pomocou troch smerovačov. Následne boli vytvorené scenáre kedy sa posielali pakety ktoré mali menšiu veľkosť MTU ako na nastavených routeroch. Potom sa posielal paket ktorý musel byť fragmentovaný ale všetky routere boli nastavené rovnako, takže fragmentácia prebehla len na klientovi. V poslednej časti sa upravili nastavenia MTU na každom routery aby bola dosiahnutá viacnásobná fragmentácia. Týmto je pre študentov viac vysvetlený princíp fragmentácie. Nedostatkom tejto úlohy je, že pri IPv6 sa fragmentuje len na koncových zariadeniach a nie na smerovačoch. Tým že smerovač nedokáže fragmentovať tieto pakety, tak posiela správu Packet Too Big. Na základe tejto správy si koncové zariadenie upraví fragmenty na požadovanú velkosť a znovu odošle. Ani súčasná verzia tohto nástroja však nedokáže spracovať tento paket. Počas simulácií som skúsil vytvoriť vlastné knižnice aby tento paket dokázali spracovať. Toto však nebola správna cesta a paket s príznakom Packet Too Big sa nepodarilo spracovať. Všetky úlohy boli prepracované do novších návrhov s rozšíreniami aby lepšie vysvetlili danú problematiku úlohy.

Literatúra

[1] AMANSINGLA. GeekforGeeks. Online. *Application Layer in OSI Model*. Dostupné z: https://www.geeksforgeeks.org/presentation-layer-in-osi-model. [cit. 2024-03-16].

[2] AMANSINGLA. GeekforGeeks. Online. *Presentation Layer in OSI model*. Dostupné z: https://www.geeksforgeeks.org/presentation-layer-in-osi-model. [cit. 2024-03-16].

[3] AMANSINGLA. GeekforGeeks. Online. *Session Layer in OSI model*. Dostupné z: https://www.geeksforgeeks.org/session-layer-in-osi-model. [cit. 2024-03-16].

[4] JavaTpoint. *Session Layer in OSI Model*. Javatpoint. Dostupné z https://www.javatpoint.com/session-layer-in-osi-model. [Citované 16.3.2024].

[5] AGARWAL, Aishwarya. GeekforGeeks. Online. *Transport Layer responsibilities*. Dostupné z: https://www.geeksforgeeks.org/transport-layer-responsibilities. [cit. 2024-04-16].

[6] GINNI. Tutorialspoint. Online. *What is the TCP Segment Header*? Dostupné z: https://www.tutorialspoint.com/what-is-the-tcp-segment-header. [cit. 2024-04-16].

[7] DWAIPAYAN_BANDYOPADHYAY. GeekforGeeks. Online. *Network Layer in OSI Model*. Dostupné z: https://www.geeksforgeeks.org/network-layer-in-osi-model. [cit. 2024-04-16].

[8] HUSTON, Geoff. APNIC. Online. *Evaluating IPv4 and IPv6 packet fragmentation*. Dostupné z: https://blog.apnic.net/2016/01/28/evaluating-ipv4-and-ipv6-packet-frangmentation. [cit. 2024-03-16].

[9] Packet Life. Online. *Path MTU Discovery*. Dostupné z: https://packetlife.net/blog/2008/aug/18/path-mtu-discovery. [cit. 2024-03-16].

[10] KANIKAJOSHI. GeekforGeeks. Online. *Data Link Layer*. Dostupné z: https://www.geeksforgeeks.org/data-link-layer. [cit. 2024-03-16].

[11] WILLIAMS, Lawrence. Guru99. Online. *Types of Computer Network: What is LAN, MAN and WAN*. Dostupné z: https://packetlife.net/blog/2008/aug/18/path-mtu-discovery. [cit. 2024-02-26].

[12] ANSHIKA GOYAL. GeekforGeeks. Online. *Types of area networks – LAN, MAN and WAN*. Dostupné z: https://www.geeksforgeeks.org/types-of-area-networks-lan-man-and-wan. [cit. 2024-03-16].

[13] Catchpoint Systems. Online. *IPv6 Multicast Address*. Dostupné z: https://www.catchpoint.com/benefits-of-ipv6/ipv6-multicast-address. [cit. 16.4.2024].

[14] Catchpoint Systems. Online. *IP Multicast*. Dostupné z: https://www.catchpoint.com/network-admin-guide/ip-multicast. [cit. 16.4.2024].

[15] M.A.K.S.M.S.A.D. *A Detailed Analogy of Network Simulators NS1, NS2, NS3 and NS4*. ijfrcsce. 2017, vol. 3, no. 12, s. 291-295. cit. 8.2.2024.

[16] PAGMAT a TOMHENDERSON. Online. In: CHANGES.md. Dostupné z: https://github.com/nsnam/ns-3-dev-git/blob/master/CHANGES.md. [cit. 2024-05-26].

[17] TARIQ, Rabia. Online. 2023. *Lecture#28 | Types of Computer Network | PAN Vs LAN Vs MAN Vs WAN | ICT*. Dotupné z: https://www.youtube.com/watch?app=desktop&v=Ehnydt2iA-c.[cit. 2024-01-19]

Zoznam symbolov a skratiek

NS3	Network Simulator version 3
IPv4	Internet Protocol version 4
IPv6	Internet Protocol version 6
ISO/OSI	International Organization for Standardization/Open Systems Interconnection
TCP/IP	Transmission Control Protocol/Internet Protocol
RTT	Round-Trip Time
TCP	Transmission Control Protocol
UDP	User Datagram Protocol
MTU	Maximum Transmission Unit
PMTUD	Path Maximum Transmission Unit Discovery
ICMP	Internet Control Message Protocol
MF	More Fragment
DF	Dont Fragment
PTB	Packet Too Big
CSMA	Carrier Sense Multiple Accessh
WAN	Wide Area Network
MAN	Metropolitan Area Network
LAN	Local Area Network
WLAN	Wireless Local Area Network
PAN	ersonal Area Network
QAM	Quadrature Amplitude Modulation

A Obsah elektronické přílohy

A.1 Virtuálny operačný systém

Táto príloha obsahuje virtuálny operačný systém Linux s nainštalovaným simulátorom a všetkými potrebnými komponentami. Prihlásenie do tohto systému vyžaduje heslo `student`.

A.2 Zdrojové kódy

Táto príloha obsahuje vypracovaná simulácie pre jednotlivé laboratórne úlohy.

A.3 Návody pre laboratórne úlohy

Súčasťou tejto prílohy sú tri dokumenty, ktoré obsahujú návody pre vypracovanie laboratórnych úloh. Tieto návody sú vypracované v českom jazyku, aby mohli byť aplikované do vyučovania.

A.4 Príloha LaTeX projektu, pre vytvorenie návodov

Aby mohli byť tieto návody v budúcnosti upravovatelné, tak boli vytvorené v programe LaTeX. V prílohe sa nachádza projekt pre ich tvorbu.