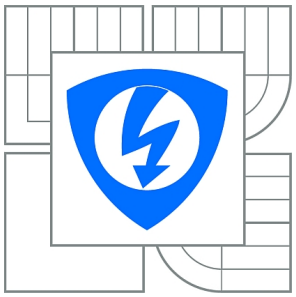
VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ

ÚSTAV TELEKOMUNIKACÍ

FACULTY OF ELECTRICAL ENGINEERING AND COMMUNICATION

DEPARTMENT OF TELECOMMUNICATIONS

MOŽNOSTI SIMULACE A OPTIMALIZACE SYSTÉMŮ HROMADNÉ OBSLUHY V PROSTŘEDÍ MATLAB

POSSIBILITIES OF SIMULATION AND OPTIMIZATION OF QUEUING SYSTEMS IN MATLAB

DIPLOMOVÁ PRÁCE

MASTER'S THESIS

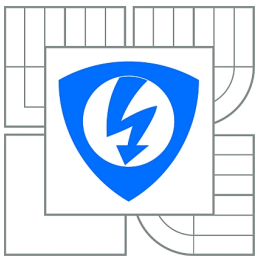
AUTOR PRÁCE — Bc. MARTIN KAKVIC

AUTHOR

VEDOUCÍ PRÁCE — Ing. ONDŘEJ KRAJSA, Ph.D.

SUPERVISOR

BRNO 2015



VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

Fakulta elektrotechniky a komunikačních technologií

Ústav telekomunikací

Diplomová práce

magisterský navazující studijní obor
Telekomunikační a informační technika

Student: Bc. Martin Kakvic ***ID:*** 151309
Ročník: 2 ***Akademický rok:*** 2014/2015

NÁZEV TÉMATU:

Možnosti simulace a optimalizace systémů hromadné obsluhy v prostředí MATLAB

POKYNY PRO VYPRACOVÁNÍ:

S využitím prostředí MATLAB navrhněte a realizujte toolbox pro návrh a realizaci ethernetových sítí, založený na popisu pomocí systémů hromadné obsluhy. Navrhněte a realizujte grafické rozhraní k tomuto sytému. Systém zaměřte především na problematiku zajištění kvality služeb v Ethernetu.

DOPORUČENÁ LITERATURA:

[1] LE BOUDEC, Jean-Yves a Patrick THIRAN. Network calculus: a theory of deterministic queuing systems for the Internet. Berlin: Springer, c2001, xix, 274 s. ISBN 3-540-42184-x.
[2] GIAMBENE, Giovanni. Queuing theory and telecommunications: networks and applications. New York: Springer, c2005, xvii, 585 s. ISBN 0-387-24065-9.
[3] ATTAWAY, Stormy. MATLAB: a practical introduction to programming and problem solving. 2nd ed. Waltham: Butterworth-Heinemann, c2012, xx, 518 p. ISBN 978-0-12-385081-2.
[4] HANSELMAN, Duane C a Bruce LITTLEFIELD. Mastering MATLAB. 1st ed. Upper Saddle River: Pearson, c2012, xiv, 843 p. ISBN 9780136013303.
[5] FRIKHA, Mounir. Ad hoc networks: routing, QoS and optimization. London: ISTE, 2011, x, 266 s. ISBN 978-1-84821-227-5.
[6] BALAKRISHNAN, Ram. Advanced QoS for multi-service IP/MPLS networks. Indianapolis: Wiley Publishing, 2008, 432 s. ISBN 978-0-470-29369-0.

Termín zadání: 9.2.2015 ***Termín odevzdání:*** 26.5.2015

Vedoucí práce: Ing. Ondřej Krajsa, Ph.D.
Konzultanti diplomové práce:

doc. Ing. Jiří Mišurec, CSc.
Předseda oborové rady

UPOZORNĚNÍ:

Autor diplomové práce nesmí při vytváření diplomové práce porušit autorská práva třetích osob, zejména nesmí zasahovat nedovoleným způsobem do cizích autorských práv osobnostních a musí si být plně vědom následků porušení ustanovení § 11 a následujících autorského zákona č. 121/2000 Sb., včetně možných trestněprávních důsledků vyplývajících z ustanovení části druhé, hlavy VI. díl 4 Trestního zákoníku č.40/2009 Sb.

Abstrakt

Táto diplomová práca sa zaoberá možnosťami simulácie a optimalizácie ethernetových sietí v prostredí Matlab. V práci je popísaná technológia Ethernet, kvality služieb a architektúra diferencovaných služieb. Na základe týchto poznatkov bol v prostredí Matlab vytvorený toolbox umožňujúci vytvorenie dynamickej siete, ktorú je možné konfigurovať pomocou grafického užívateľského rozhrania. V závere práce sú uvedené výsledky simulácie sietí, ktorá prebehla vo vytvorenom programe.

Kľúčové slová

Ethernet, Matlab, simulovanie, kvalita služby, grafické uživateľské rozhranie, diferencované služby

Abstract

This master thesis deals with the possibilities of simulation and optimalization of ethernet networks in Matlab interface. Ethernet technology, quality of service and architecture of differentiated services are described in this thesis. Based on these facts, there was created a toolbox. This toolbox allows to create a dynamic network, which can be configured via using GUI. In the end of the thesis, there are shown results of the simulation of networks, which is running in created program.

Key words

Ethernet, Matlab, simulation, quality of service, graphic user interface, DiffServ

KAKVIC, M. *Možnosti simulace a optimalizace systémů hromadné obsluhy v prostředí MATLAB.* Brno: Vysoké učení technické v Brně, Fakulta elektrotechniky a komunikačních technologií, 2015. 44 s. Vedoucí diplomové práce Ing. Ondřej Krajsa, Ph.D..

PREHLÁSENIE

Prehlasujem, že som svoju diplomovú prácu na tému „Možnosti simulace a optimalizace systémů hromadné obsluhy v prostředí MATLAB“ vypracoval samostatne pod vedením vedúceho diplomovej práce, využitím odbornej literatúry a ďalších informačných zdrojov, ktoré sú všetky citované v práci a uvedené v zozname literatúry na konci práce. Ako autor uvedenej diplomovej práce ďalej prehlasujem, že v súvislosti s vytvorením tejto diplomovej práce som neporušil autorské práva tretích osôb, najmä som nezasiahol nedovoleným spôsobom do cudzích autorských práv osobnostných a/nebo majetkových a som si plne vedomý následkov porušenia ustanovenia S 11 a nasledujúcich autorského zákona č. 121/2000 Sb., o právu autorském, o právach súvisejúcich s právom autorským a o zmene niektorých zákonov (autorský zákon), vo znení neskorších predpisov, vrátane možných trestnoprávnych dôsledkov vyplývajúcich z ustanovenia časti druhé, hlavy VI. diel 4 Trestného zákoníka č. 40/2009 Sb.

V Brne dňa: . . . . . podpis autora: . . . . . .

Poďakovanie

Touto cesto by som rád poďakoval vedúcemu diplomovej práce Ing. Ondřej Krajsa, Ph.D. za odborné rady a pripomienky, ktoré mi pomohli pri písaní bakalárskej práce.

Faculty of Electrical Engineering
and Communication
Brno University of Technology
Technicka 12, CZ-61600 Brno
Czech Republic

http://www.six.feec.vutbr.cz

PODĚKOVÁNÍ

Výzkum popsaný v této diplomové práci byl realizován v laboratořích podpořených z projektu SIX; registrační číslo CZ.1.05/2.1.00/03.0072, operační program Výzkum a vývoj pro inovace.

Brno . . . . . .

. . . . . .
(podpis autora)

EVROPSKÁ UNIE
EVROPSKÝ FOND PRO REGIONÁLNÍ ROZVOJ
INVESTICE DO VAŠÍ BUDOUCNOSTI

OBSAH

ZOZNAM OBRÁZKOV

ÚVOD

Matlab ako programové prostredie patrí medzi programy, ktoré sa špecializujú na riešenie a modelovanie rôznych matematických úloh. Či už sa jedná o spracovanie signálov, obrazu alebo iných, Matlab poskytuje širokú škálu funkcií. Medzi tieto funkcie patria napr aj tzv. t*oolboxy. Toolbox* je kolekcia niekoľkých programových funkcií, ktoré umožňujú testovať a modelovať mnohé scenáre spojené s funkcionalitou daného *toolboxu.* Podstata môže spočívať aj v tom, že si na počítačovej simulácií namodelujeme reálny systém a sledujeme jeho správanie sa pri rôznych vstupných parametroch. Výsledky týchto experimentálnych testovaní vieme spätne aplikovať na reálny systém, napr. jeho zoptimalizovaním.

Konkrétnym príkladom testovania môžu byť aj IP siete, ktoré prenášajú dátové toky rôznych služieb. V súvislosti s vysokým rozmachom internetu a služieb ním poskytovaným sú poskytovatelia nútení ponúkať svoje služby vo vysokej kvalite, teda s čo najmenším oneskorením, vysokými prenosovými rýchlosťami a určitými garanciami kvalít. To, aby mohli byť tieto služby poskytované v určitej kvalite, je nutné zabezpečiť. Samozrejme, dohliadanie na tieto služby nemôže byť vykonávané manuálne, pretože by to bolo nefektívne, a preto je potrebné implementovať automatizované systémy.

Cieľom tejto práce je navrhnutie a realizácia *toolboxu* pre návrh a realizáciu *ethernetových* sietí. Ďalšou úlohou bol návrh a realizácia grafického rozhrania pre takýto systém, pričom bolo potrebné zabezpečiť kvalitu služieb v *Ethernete*.

Táto práca je členená do piatich kapitol. Prvá časť sa zaoberá technológiou *Ethernet*, druhá časť sa venuje kvalite služieb a jej parametrom. Ďalšia kapitola sa pojednáva o architektúre diferencovaných služieb, pričom je podrobnejšie popísané pole TOS, nástroje zaistenia kvality a metódy radenia paketov do čakacích radov. Predposledná kapitola obsahuje informácie o realizácii *toolboxu* v programovacom prostredí Matlab a bližšie popisuje jednotlivé súčasti *toolboxu.* Posledná kapitola je venovaná vytvorenému grafickému užívateľskému rozhraniu a popisuje hlavný panel a jeho časti a simuláciu.

1. ETHERNET

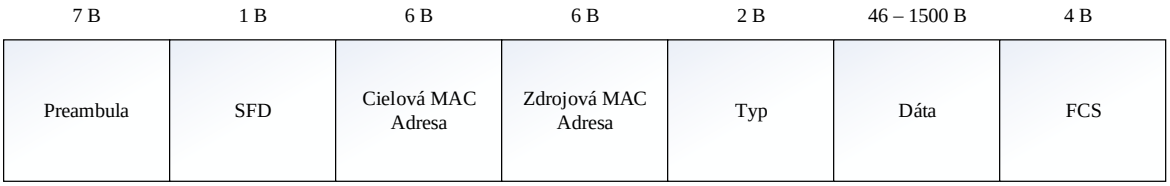
echnológia *Ethernet* je v súčasnej dobe najrozšírenejšou sieťovou technológiou. Jej počiatky siahajú do 70. rokov 20. storočia do výskumných laboratórií spoločnosti Xerox PARC. Prototyp vtedajšieho *Ethernetu* dosahoval prenosovú rýchlosť 2,94 Mbit/s na koaxiálnom kábli s impedanciou 70 Ohm. Neskôr sa o túto technológiu začali zaujímať aj firmy ako Intel a DEC. Spoločne sa pokúšali o vytvorenie vylepšenej verzie *Ethernetu* s názvom DIX Ethernet. Spolupráca týchto firiem vyústila k tomu, že ďalšiu štandardizáciu prenechali skupine IEEE (*Institute of Electrical and Electronics Engineers*), konkrétne pracovnej skupine 802.3. Výsledkom ich práce bol prvý štandard z roku 1985 pomenovaný ako IEEE 802.3 „*Carrier sense Multiple Acess with Collision Detection* (CSMA/CD) *access method and physical layer specification*". Keďže názov *Ethernet* si nechala zaregistrovať spoločnosť Xerox, štandard definovaný inštitúciou IEEE sa nenazýval *Ethernetom,* ale metódou CSMA/CD.

Významnou črtou *Ethernetu* je jeho prístupová metóda, ktorá rieši prístup uzlu k prenosovému médiu. Jedná sa o už spomenutú metódu CSMA/CD, ktorá sa používa pri prístupe na zdieľanom médiu a zabezpečuje, aby nevysielali viaceré uzly v jednom momente. Môžeme povedať, že metóda obsahuje 3 základné princípy, z ktorých sa skratka CSMA/CD skladá. Prvým je CS (*Cerrier Sense*), ktorý vyjadruje to, že ak niektorý z uzlov chce vysielať, musí najprv naslúchať a zistiť, či nevysiela v tom momente niekto iný. Ak zistí, že žiadny z uzlov v danej chvíli nevysiela, môže začať vysielať on. Druhým s princípov je MA (Multiple *Access*), ktorý umožňuje viacnásobný prístup uzlov k médiu. Inak povedané, možnosť, aby vysielali viaceré uzly. Posledným je CD (*Collision Detection*), ktorý má na starosti zisťovanie kolízií. V tom istom čase, ako uzol vysiela, naslúcha k tomu, či nevysiela aj iný z uzlov. V prípade, že áno, ukončí svoje vysielanie a odošle tzv. JAM signál, ktorý informuje o tom, že došlo ku kolízii. Následne je stanica na istú dobu odmlčaná a čaká na opätovné vysielanie[1].

Postupom času sa však zistilo, že metóda CSMA/CD síce spĺňa svoju úlohu, no z pohľadu ďalšieho vývoja nie je až taká výhodná ako využitie prepínačov, a teda zavedenie plne duplexného prenosu informácií.

1.1 Ethernet rámec

Dátová jednotka operujúca na linkovej vrstve ISO/OSI technológie *Ethernet* sa nazýva rámec. *Ethernet* rámec slúži na zapuzdrenie paketov, ktoré linkovej vrstve predala sieťová vrstva. Každý z týchto rámcov má rovnakú štruktúru, ktorú možno vidieť na nasledujúcom obrázku 1.1.



Obr. 1.1.: Ethernet rámec

- **Preambula** – toto pole obsahuje striedajúce sa jednotky a nuly definujúce hodinový signál o hodnote 5 MHz na začiatku každého rámca. Teda oznamuje prijímaču, že sú mu odosielané rámce.
- **SFD (*Start of Frame Delimiter*)** – úvodný oddeľovač rámca, ktorý má hodnotu jedného Bytu, a to 10101011. Na základe tejto hodnoty, konkrétne poslednej dvojice jednotiek, sa môže príjemca synchronizovať a detekovať tak začiatok dát aj v prípade, že začne naslúchať uprostred.
- **Cieľová MAC adresa** – už samotný názov pola hovorí o význame. Podľa tejto adresy funguje na druhej vrstve prepínanie rámcov, alebo teda ktorým portom na zariadení má byť rámec odoslaný. Cieľová adresa môže byť adresa jedného uzlu, všesmerová alebo viacsmerová.
- **Zdrojová MAC adresa** – jedná sa o adresu zariadenia, ktoré rámec odosiela. V tomto prípade môže byť adresa iba adresou jedného uzlu. MAC adresa je ďalej unikátnym identifikátorom každého sieťového rozhrania. Je fyzicky pridelená výrobcom daného sieťového rozhrania. MAC adresa má veľkosť 48 bitov a jej zápis môže vyzerať nasledovne: *00:0A:AC:9D:B1:48*.

- **Typ** – ak je táto hodnota menšia ako 0x600 v hexadecimálnej sústave, určuje dĺžku rámca. V opačnom prípade určuje použitý protokol vyššej vrstvy, teda sieťovej.
- **Dáta** – veľkosť tohto pola určuje veľkosť paketu, ktorý bol predaný sieťovou vrstvou.
- **FCS (*Frame Check Sequence*)** – kontrolný súčet, ktorý slúži na detekciu chýb v rámci[1].

1.2 Komunikačné štandardy Ethernetu

V súvislosti so zavedením *Ethernetu* ako sieťového štandardu sa vytvorili jeho určité triedy. Jedná sa o to, že jednotlivé triedy majú rôzne prenosové rýchlosti, útlm, najdlhšiu možnú vzdialenosť alebo druh použitého prenosového média[2].

***Ethernet* – 10BASE** disponuje najvyššou možnou prenosovou rýchlosťou 10 Mb/s.

- V prípade 10BASE5 (*thick Ethernet*) sa ako fyzické prenosové médium používa hrubý koaxiálny kábel, ktorý je schopný prepojiť 100 koncových staníc na maximálnu vzdialenosť 500 m.
- 10BASE2 zabezpečuje prenos po tenkom, dvakrát tienenom koaxiálnom kábli o maximálne dĺžke 200 m a umožňuje pripojenie 30 koncových staníc.
- 10BASE-T, známy tak isto ako *twisted pair* (krútená dvojlinka), umožňuje vzdialenosť do 100 m netieneným TP káblom.
- 10BASE-FL používa optické vlákna ako prenosové médium na maximálnu vzdialenosť 2 km.

***Fast Ethernet* – 100BASE** pracuje s maximálnou prenosovou rýchlosťou 100 Mb/s.

- 100BASE-TX používa ako prenosové médium krútenú dvojlinku kategórie 5 využívajúcu 2 páry. Realizovateľná vzdialenosť je do 200 m.
- 100BASE-FX ako prenosové médium používa multimódový optický kábel. Použitie je možné do vzdialenosti 2 km.
- 100BASE-T4 ako prenosové médium používa krútenú dvojlinku kategórie 3 a využíva 4 páry do vzdialenosti 100 m.

***Gigabit Ethernet* – 1000BASE** umožňuje prenosovú rýchlosť do 1 Gb/s.

- 1000BASE-T využívajúci 4 páry z krútenej dvojlinky kategórie 5 do vzdialenosti 100 m.
- 1000BASE-SX,LX používa ako prenosové médium optický kábel, zabezpečujúci v prípade SX spojenie do vzdialenosti 550 m pri použití multimódu. V prípade LX a jedného módu je vzdialenosť 5 km a pre multimód rovnako ako v predošlom prípade 550 m.

***10 Gigabit Ethernet* – 10GBASE** umožňuje prenosovú rýchlosť do 10 Gb/s. Jedná sa o riešenie využívajúce od lokálnych sieti, cez metropolitné až po globálne siete. Štandard bol organizáciou IEEE označený ako 802.3ae3

- 10GBASE-SR a 10GBASE-Sw využívajú multimódové optické vlákna s krátkymi vlnovými dĺžkami 850 nm. Využitie je možné do vzdialenosti 200 m.
- 10GBASE-LR a 10GBASE-LW využívajú jednomódové optické vlákna s dlhými vlnovými dĺžkami, t.j. 1310 nm. Využitie je umožnené do vzdialenosti 10 km.
- 10GBASE-ER a 10GBASE-EW opäť využívajú jednomódové optické vlákna pracujúce s vlnovou dĺžkou 1550 nm. Veľkou výhodou tohto štandardu je, že umožňuje vzdialenosť až 40 km a bez použitia aktívneho prvku.
- 10GBASE-LX4 umožňuje použitie multimódového aj jednomódového optického vlákna pri vlnovej dĺžke 1310 nm. V prípade multimódového je možné prepojiť stanice vo vzdialenosti 300 m a pri jednomódovom 10 km.

***100 Gigabit Ethernet* – 100GBASE** svojou architektúrou poskytuje vysokú škálovateľnosť a je dobrým pre plnenie potrieb poskytovateľov služieb. Jeho štandardizácia bola povolená v roku 2010 organizáciou IEEE pod označením 802.3ba. Možnosť podporovať vysoké rýchlosti sa v tomto prípade rieši takým spôsobom, že komunikačný tok sa rozdelí medzi jednotlivé bloky, a tie do niekoľkých optických káblov. Týmto je možné pre prenosovú kapacitu 100 Gb/s využiť desať 10 Gb/s kanálov.

- 100GBASE-CR10 prenos zabezpečuje koaxiálnym káblom, tzv. *Twin-ax*. Jedná sa o moderný typ koaxiálneho kábla, kde sú vo vnútri použité 2 vodiče. Týmto je možné dosiahnuť vysoké prenosové rýchlosti pre malé vzdialenosti, a to 10 m.

- 100GBASE-SR10 používa ako prenosové médium multimódové optické vlákno s vlnovými dĺžkami 850 nm. Vzdialenosť, do ktorej je možné prepojiť stanice, je 100 m.
- 100GBASE-LR4/ER4 štandardy sú veľmi podobné. Oba používajú ako prenosové médium optický kábel pri vlnovej dĺžke 1310 nm. Rozdielom je však maximálna vzdialenosť, a to pri LR4 maximálne 10 km a pri ER4 je to až 40 km.

2. KVALITA SLUŽBY

V posledných rokoch nastal obrovský nárast kapacity sietí, a taktiež aj nárast využitia tejto kapacity. Sieťová prevádzka hlasu, videa, sťahovania súborov, prehliadania stránok a iných aplikácií cez IP siete má významnú úlohu pri zvýšení prevádzky v našich sieťach. Niektoré aplikácie, ako napríklad interaktívne multimédia, nepotrebujú len veľkú šírku prenosového pásma, ale aby sieť poskytovala špeciálne služby, ktoré venujú pozornosť oneskoreniu a strate paketov.

Sieť musí mať dostatočnú šírku prenosového pásma, aby mohla čeliť požiadavkám zaťaženia. Dokonca môže vzniknúť rozpor aj pri krátkych časových intervaloch pre poskytnutie určitej šírky prenosového pásma. Toto môže nastať, pretože prevádzka v sieti nie je šírená rovnomerne počas celej doby. Ak sieťové zariadenie príjme istý typ prevádzky, tak ho spracuje, a pošle ďalej. Množstvo prevádzky, ktoré môže sieťové zariadenie posielať ďalej, je limitované kapacitou jeho pripojenia. Ak nastane prípad, kedy príde veľké množstvo paketov k zariadeniu v jednom okamihu, zariadenie nemôže pakety ihneď ďalej poslať, pretože má obmedzenú kapacitu. Pakety, s ktorými zariadenie nepracovalo, sú buď zahodené, alebo čakajú vo fronte. Toto spôsobuje preťaženie siete, čoho výsledkom je oneskorenie alebo strata paketov. Na riešenie tohto problému existuje niekoľko mechanizmov, ktoré budú popísané v ďalších kapitolách.

2.1 Parametre kvality služby

Nasadenie mechanizmov kvality služby (*Quality of Service* - QoS) má v úmysle poskytovať určité výkonnostné hranice pripojenia do siete. Šírka prenosového pásma, strata paketov, oneskorenie a kolísanie oneskorenia sú bežne používané parametre na charakterizáciu výkonnosti siete.

2.1.1 Šírka prenosového pásma

Šírka prenosového pásma je definovaná ako priepustnosť daného média, protokolu alebo siete. Inak povedané, je to prenosová rýchlosť dát, ktorou sa šíria pakety cez sieť. Je známe, že čím máme väčšiu šírku prenosového pásma, tým rýchlejšie sa pakety môžu sieťou šíriť. Vo všeobecnosti, dátové alebo komunikačné spojenia požadujúce garantované služby, majú určité požiadavky na šírku prenosového pásma, a preto potrebujú od siete aby im pridelila minimálnu potrebnú šírku prenosového pásma konkrétne pre dané spojenie. Napr. nech digitalizované hlasové aplikácie vytvárajú 64 kbit/s dátový tok. Ak by v tomto prípade sieť danému spojeniu pridelila menšiu šírku prenosového pásma, aplikácia by sa stala nepoužiteľnou [3].

2.1.2 Stratovosť

Tento parameter definuje počet paketov, ktoré boli počas prenosu od vysielača k prijímaču stratené. Strata paketov môže byť zapríčinená preťažením siete, alebo zlou kvalitou danej siete. Pod preťažením je možné rozumieť vyťaženie procesora sieťového zariadenia, alebo pretečením vyrovnávacej pamäte, atď. Stratovosť je možné vypočítať nasledujúcim vzťahom :

$$Stratovosť = \frac{Počet\ odoslaných\ paketov - Počet\ prijatých\ paketov}{Počet\ odoslaných\ paketov}. 100\%\ [\%] \quad (2.1)$$

2.1.3 Oneskorenie

Oneskorenie je čas, ktorý uplynie za dobu, kým odoslaný paket dorazí od odosielateľa k príjemcovi. V kontexte hlasových služieb je oneskorenie čas medzi vyslovením slova a jeho doručením príjemcovi. Toto oneskorenie býva spôsobené viacerými faktormi, ktoré sú prítomné pri digitalizovaní hlasu, jeho prenose cez sieť a následným dekódovaním na strane príjemcu. Rozlišujeme tieto typy oneskorení [4]:

- Paketizačné oneskorenie – vzniká na koncových bodoch linky. Patrí sem oneskorenie kódovacieho zariadenia a oneskorenie spôsobené zapuzdrovaním, napr. hlasových vzoriek do paketov.

- Prístupové oneskorenie – jedná sa o oneskorenie spôsobené radením paketov do výstupných front sieťových uzlov a následným sériovým odosielaním na linku. Je závislé od dĺžky paketov a aktuálneho vyťaženia prenosovej linky. Pre vybrané dátové prenosy je možné ho znížiť radením do prioritného frontu.

- Prenosové oneskorenie – je to čas, za ktorý je paket prenesený cez linku. Toto oneskorenie je závislé na dĺžke prenosového média a na jeho fyzikálnych vlastnostiach.

2.1.4 Kolísanie oneskorenia

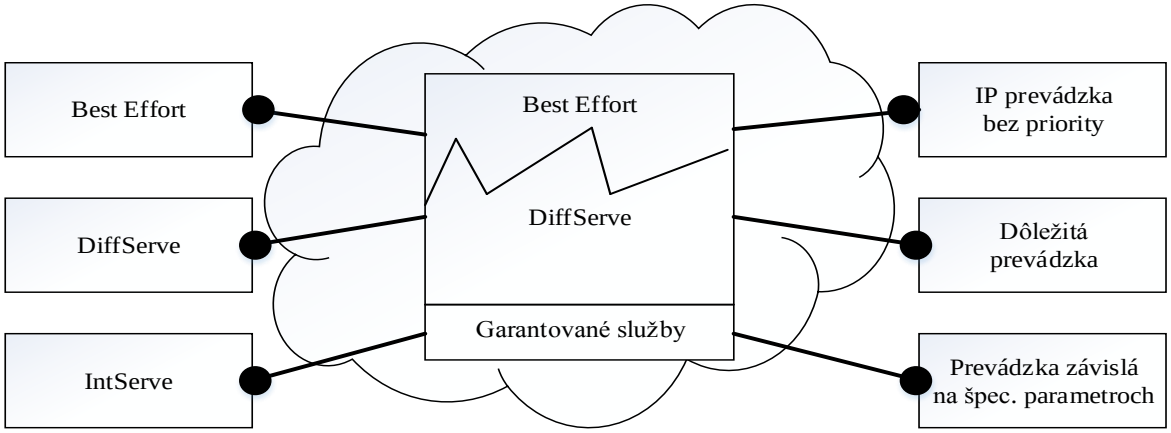
Kolísanie oneskorenia je definované ako časový rozdiel medzi dvomi po sebe prijatými paketmi. Na strane odosielateľa sú pakety odosielané rovnomerne ako súvislý dátový tok. Vplyvom zahltenia v sieti, alebo nesprávneho radenia do frontov, dochádza k vzniku nepravidelných časových medzier medzi paketmi, ktoré sa nepriaznivo prejavia na výslednom prenose. Na elimináciu tohto javu sa na strane príjemcu používa vyrovnávacia pamäť, do ktorej sa ukladajú prijaté pakety a v prípade dlhšieho čakania na prichádzajúci paket sa odošlú kódovaciemu zariadeniu. Týmto sa zamedzí vzniku hluchých miest v hlasovom prenose. [4]

2.2 Prehľad metód zabezpečenia kvality služby

Z dôvodov, ako sú napr. nemožnosť rozširovania sieti z technicko-ekonomického hľadiska alebo nutnosť použitia existujúcej infraštruktúry, je kvalitu služby možné zabezpečiť jedine na existujúcej sieti so zachovaním použitých sieťových prvkov. Metódy, ktorými je toto možné dosiahnuť, sú tzv. Diferencované a Integrované služby. Aplikácie, ktoré sú kritickými, je v dnešnej dobe mnoho. No najviac citlivými sú aplikácie, ktoré fungujú v reálnom čase, a to napr. hlasové služby alebo videokonferencia. Mechanizmy QoS možno teda rozdeliť do troch skupín:

- *Best effort* služby
- Diferencované služby
- Integrované služby

Aby bola požadovaná kvalita služby dosiahnutá v plnej miere, je dôležité, aby všetky prvky siete na prenosovej ceste podporovali dohodnuté mechanizmy.



Obr 2.1.: Hierarchia QoS

2.2.1 *Best Effort*

Model *Best-Effort* je výsledkom štandardnej funkcionality sieťových uzlov, ktoré implicitne odosielajú prichádzajúce pakety v poradí, v akom boli prijaté. Tento model je vhodný iba pre služby, ktoré nevyžadujú garanciu kvality prenosu. V prípade zahltenia budú pakety zahadzované bez ohľadu na ich dôležitosť. Za účelom toho, aby nedochádzalo k stratovosti alebo oneskoreniu paketov pre aplikácie, ktoré sú kritické, boli vyvinuté metódy *DiffServ* a *IntServ*.

2.2.2 Integrované služby (*IntServ*)

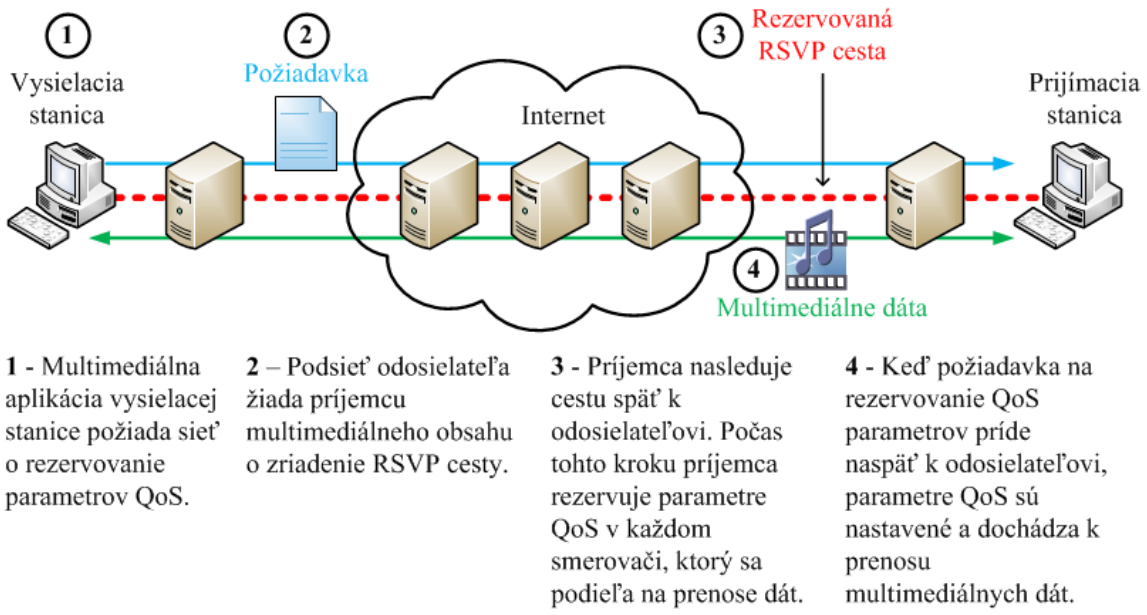
Intergrované služby sú výsledkom práce organizácie IETF, ktorý je uvedený v dokumente RFC 1633 a pochádza z roku 1994. Spolu s Integrovanými službami je definovaný dôležitý protokol RSVP. Nosnou myšlienkou Integrovaných služieb je rezervovanie sieťových parametrov pre konkrétne dátové toky. Presne pridelené a rezervované prenosové kapacity majú zaručiť stabilné prenosové vlastnosti počas celej doby trvania výmeny multimediálnych dát. Zabezpečenie QoS pomocou integrovaných služieb v mnohých aspektoch pripomína okruhovo prepínanú telefóniu.

V tomto kontexte by sme skôr spomenuli nevýhodu súvisiacu s tým, že musí byť permanentne zriadený prenosový kanál, a tak dochádza k plytvaniu prenosových prostriedkov siete pre iné aplikácie. Ďalšou nevýhodou je, že každé zo zariadení na prenosovej ceste musí mať informácie o rezervovaných dátových tokoch, a tým sa zvyšujú aj nároky na pamäť. Toto by bolo pri veľkých sieťach nepredstaviteľné, aby si jednotlivé zariadenia pamätali informácie o celej prevádzke, a tak je teda efektívnejšie nasadiť *IntServ* do menších sietí[5].

RSVP

Je to signalizačný protokol, ktorý sa používa na požadovanie prostriedkov zo siete. Môže žiadať o unicastové alebo multicastové toky. *RSVP* umožňuje aplikácii komunikovať o ich prevádzkovom profile a žiadať špecifickú kvalitu služby zo siete. *RSVP* správy sú nútené ísť cez sieť tou istou cestou, pretože *RSVP* môže rezervovať prostriedky v tom istom zariadení, ktoré bude spracovávať prevádzkové toky.

Obyčajne sú *RSVP* správy generované aplikáciami používateľa. Táto správa obsahuje cieľovú adresu, adresu zdroja, port a protokol. Ďalej obsahuje presný profil prevádzky a žiadosti o príslušných triedach služby pre daný profil. Ak sieťové zariadenie príjme žiadosť o rezervovanie, porovná túto požiadavku s dostupnými prostriedkami. Zariadenie môže svoje rozhodnutie zakladať aj na zásadách. Ak zariadenie nemá dostatočné prostriedky vyhovieť požiadavke, tak ju zamietne a žiadajúcej aplikácií pošle chybovú správu. Na druhej strane, ak sa zariadenie rozhodne prideliť požiadavku, informácia o rezervovaní je nainštalovaná do zariadenia a RSVP správa je predaná ďalšiemu zariadeniu na ceste. Ak budú mať všetky zariadenia na ceste rezervované prostriedky, koncové zariadenie si môže byť isté, že prostriedky, o ktoré žiadalo, sú pre jeho účel rezervované.



Obr 2.2.: Proces vytvorenia rezervovaného kanála pomocou protokolu RSVP

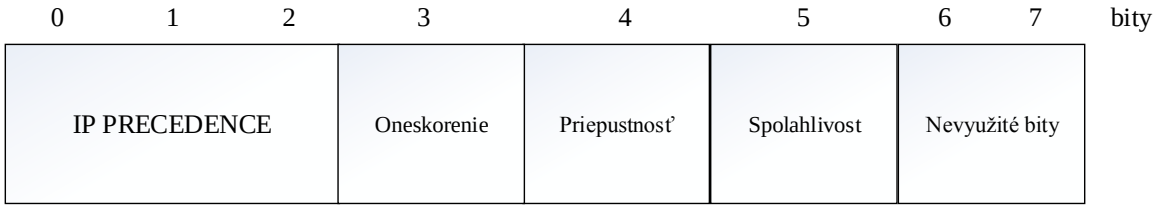
2.2.3 Diferencované služby (*DiffServ*)

Táto architektúra ponúka ďalší pohľad na zabezpečenie kvality služby v IP sieťach. Je definovaná v rámci RFC 2475 pochádzajúceho z roku 1998. Na rozdiel od *IntServ* a *DiffServ* nerezervuje celý dátový tok medzi komunikujúcimi uzlami, ale vytvára niekoľko tried prevádzky. Na základe toho, do ktorej triedy príslušný paket prislúcha, je s ním zaobchádzané. Každé zariadenie, s ktorým príde paket do konktatu na prenosovej ceste, bude informované o dôležitosti daného paketu. Mechanizmy zaobchádzania s jednotlivými paketmi budú popísané v nasledujúcej časti. Medzi nevýhody *DiffServ* patri napríklad, že nie je možné zaručiť kvalitu služby na celej prenosovej trase, pretože nie každé zariadenie musí podporovať túto službu. Ich výhodou je, že jednotlivé zariadenia si nemusia pamätať údaje o jednotlivých dátových tokoch, a tak isto nie je zbytočne rezervované prenosové pásmo. Vo výsledku by sa dalo povedať, že hlavným rozdielom medzi *IntServ* a *DiffServ* je, že *IntServ* pristupuje ku každému dátovému toku jednotlivo a *DiffServ* združuje dátové toky do skupín, pre ktoré poskytuje QoS hromadne. V rámci tejto práce boli pre účely QoS zavedené diferencované služby[6].

3. ARCHITEKTÚRA DIFERENCOVANÝCH SLUŽIEB

3.1 Význam hlavičky TOS

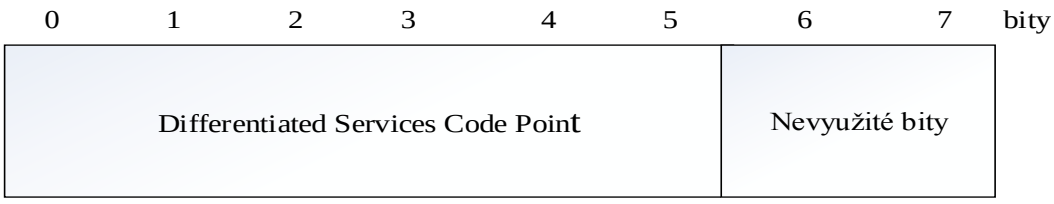
Ako už bolo spomenuté, *DiffServ* združuje dátové toky do skupín a následne im prideľuje priority. Kategorizácia paketov je realizovaná pomocou poľa TOS (*Type of Service*) v hlavičke paketu IP protokolu verzie 4. Náhľad do poľa TOS môžeme vidieť na nasledujúcom obrázku 3.1.



Obr. 3.1.: TOS hlavička RFC 791

Podľa štandardu RFC 791 boli prvé tri bity definované na klasifikáciu paketov, ktorá umožňovala pakety rozdeliť do 8 tried podľa priority. Ďalšie 3 bity boli využité na zadefinovanie priority pre oneskorenie, priepustnosť a spoľahlivosť. Zvyšné dva bity ostali nevyužité. Štandardom RFC 1349 došlo k predefinovaniu jedného z dvojice nevyužitých bitov tým spôsobom, že bol využitý na účel rozlišovania nákladov na prenos paketu. Tento model je známy aj pod menom *IP Precedence* a je predchodcom súčasného modelu *DiffServ*. Nevýhodou *IP Precedence* bol nízky počet možných tried prevádzky, nedefinovaná priorita zahadzovania paketov v rámci triedy a nekonzistentná implementácia výrobcami sieťových zariadení.

Model *DiffServ* definovaný v štandarde RFC 2474 zmenil využitie poľa ToS za účelom zvýšenia počtu možných tried prevádzky a odstránenia nevýhod *IP Precedence*. Na nasledujúcom obrázku sú znázornené jednotlivé polia DSCP (*Differentiated Services Code Point*) [8].



Obr. 3.2.: Pole hlavičky DSCP

DSCP hlavička sa skladá z 8 bitov, pričom je využitých 6 z nich. Zvyšné dva nevyužité bity majú podľa RFC 3169 označenie *Explicit Congestion Notification.* Na základe 6 bitov, ktoré sú využívané, je možné zadefinovať 64 tried prevádzky. Pakety, ktoré majú rovnaké označenie DSCP, tvoria spoločnú skupinu správania (*Behavior Aggregate*), v rámci ktorej sa im poskytuje QoS. Keďže pri *DiffServ* rozhoduje každé zariadenie samostatne ako bude zaobchádzať s jednotlivými triedami prenosu hovoríme o tzv. správaní sa jednotlivých sieťových uzlov (*Per Hop Behavior -PHB*). Pod týmto pojmom rozumieme klasifikáciu paketov do treid, ich radenie do čakacích radov, dohliadanie a iné [7,8]. Existuje niekoľko štandardizovaných PHB správaní, ktoré sú uvedené v nasledujúcich podkapitolách.

3.1.1 Predvolené PHB správanie

Predvolené PHB (*Default PHB*) správanie špecifikované v RFC 2474 definuje, že pakety označené DSCP kódom 000000 budú sieťovými zariadeniami spracované podľa modelu *Best-effort* a nebudú mapované do žiadnej inej triedy prenosu [7].

3.1.2 Selektor triedy

Selektor triedy (*Class Selector*) je PHB správanie definované v RFC 2474 za účelom zachovania kompatibility s modelom *IP Precedence*. Na klasifikáciu paketov využíva iba prvé tri bity poľa DSCP. Zvyšné tri bity necháva nastavené na hodnotu nula. Selektor triedy zachováva rovnaké PHB správanie ako bolo definované v *IP Precedence* [7].

3.1.3 Urýchlené odosielanie

Urýchlené odosielanie (*Expedited Forwarding - EF*) je PHB správanie definované v RFC 2598 s cieľom poskytnúť vysokú mieru kvality prenosu pre kritické aplikácie, ako napríklad VoIP. Toto správanie môže byť na sieťových uzloch implementované formou radenia prevádzky do prioritného frontu. Urýchlené odosielanie poskytuje nízku stratovosť paketov, garantovanú šírku pásma, nízke oneskorenie a jeho kolísanie [7].

3.1.4 Zaručené odosielanie

Zaručené odosielanie (*Assured Forwarding – AF*) je PHB správanie, ktoré umožňuje rozdelenie prevádzky do štyroch tried (AF1 až AF4), pričom v rámci každej triedy existujú tri stupne zahadzovania paketov. Bolo definované v RFC 2597. Rozdelenie tried a priorít zahadzovania môžeme vidieť v nasledujúcej tab. 3.1[8].

Priorita zahadzovania	Trieda 1	Trieda 2	Trieda 3	Trieda 4
Nízka	001010 AF11 DSCP 10	010010 AF21 DSCP 18	011010 AF31 DSCP 26	100010 AF41 DSCP 34
Stredná	001100 AF12 DSCP 12	010100 AF22 DSCP 20	011100 AF32 DSCP 28	100100 AF42 DSCP 36
Vysoká	001110 AF13 DSCP 14	010110 AF23 DSCP 22	011110 AF33 DSCP 30	100110 AF43 DSCP 38

Tab. 3.1.: Triedy a priority zahadzovania paketov v modeli DiffServ

3.2 Nástroje pre zaistenie QoS

Za účelom dosiahnutia kvality služby v prípade diferencovaných služieb existuje niekoľko iných techník, ktoré budú popísané v nasledujúcej časti.

3.2.1 Klasifikácia paketov

Na vstupe do dátovej siete je každý IP paket označený značkou, ktorá určuje, ako sa má s daným paketom zaobchádzať. Inak povedané, určuje „triedu prenosu“ poskytnutú paketu. Počas ďalšieho prenosu paketov dátovou sieťou ostatné zariadenia iba prečítajú značku každého paketu, na základe ktorej sa riadia pri zaobchádzaní s paketom. Toto označenie je uložené priamo v hlavičke IP paketu, a to v poli TOS [9].

3.2.2 Označovanie paketov

Označovanie paketov je úzko spojené s klasifikáciou paketov a umožňuje určiť triedy prenosov podľa ich priority. Najčastejšie je realizované na hranici siete, kde sú paketom nastavené hodnoty DSCP, *IP Precedence*, alebo iné v zhode s triedami použitými vo vnútri siete. Zariadenia v sieti potom nemusia opätovne klasifikovať pakety na základe rôznych vrstiev TCP/IP, čo vo väčšine prípadov ani nie je možné, ale použijú hodnotu klasifikačného poľa[9].

3.2.3 Dohliadanie a tvarovanie prevádzky

Dohliadanie a tvarovanie sú najstaršími QoS mechanizmami. Monitorujú využitie šírky pásma a snažia sa ju limitovať na požadovanú hodnotu. Dohliadanie reaguje na prekročenie limitu okamžitou reakciou, ktorá môže znamenať zmenu značenia paketov alebo ich zahodenie. Tvarovanie pracuje s mechanizmami radenia paketov do frontov, kam dočasne ukladá pakety v čase zahltenia. Pakety sú teda zahadzované až v prípade vysokej miery zahltenia, keď veľkosť výstupného frontu nepostačuje na uchovanie potrebného množstva paketov. Čakanie vo výstupnom fronte môže znamenať zvýšenie oneskorenia a jeho kolísania, preto je napr. pre VoIP aplikácie vhodnejšie použiť dohliadanie prevádzky. [9]

3.2.4 Plánovanie a radenie paketov

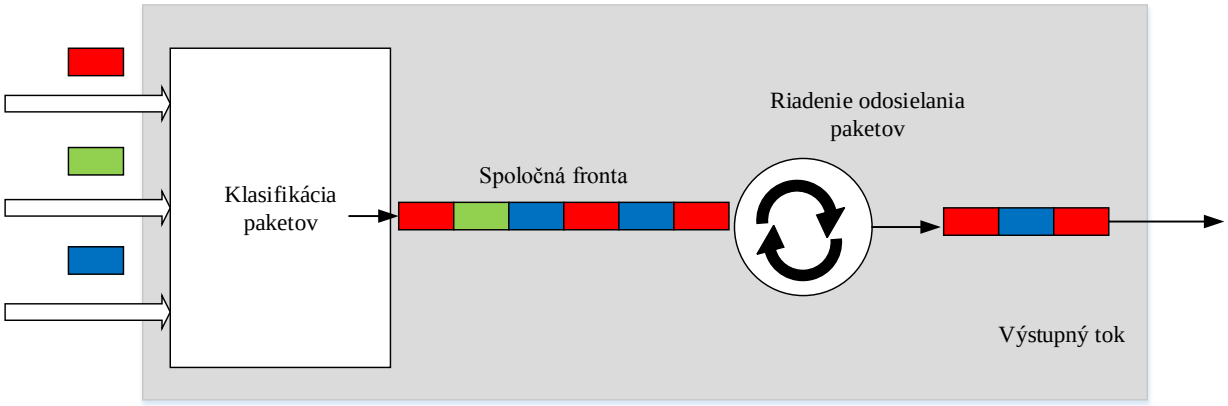
Plánovanie je proces rozhodovania, ktorý paket bude odoslaný ako ďalší v poradí. Tento proces je prítomný pri odosielaní každého paketu, aj keď výstupné rozhranie nezaznamenáva zahltenie. V prípade nízkeho vyťaženia linky sú pakety odosielané v poradí v akom prichádzajú. Ak nastane zahltenie, uplatnia sa nástroje radenia paketov do čakacích radov, známe tiež ako nástroje na riadenie zahltenia. Tieto mechanizmy sú aktívne len ak množstvo prichádzajúcej sieťovej prevádzky prevyšuje prenosovú kapacitu výstupného rozhrania. Plánovanie je prítomné aj v čase zahltenia, kedy plánovač musí rozhodnúť, v akom poradí a ako často bude obsluhovať jednotlivé fronty.

3.3 Metódy radenia paketov do čakacích radov

V tejto kapitole budú bližšie popísane štyri metódy radenia paketov do čakacích radov [10].

3.3.1 FIFO (*First In First Out*)

Svojou funkciou sa zaraďuje medzi najjednoduchší základný mechanizmus so zachovaním poradia požiadaviek. Uvoľňovanie linky prebieha postupne, podľa poradia príchodu jednotlivých paketov. Z tohto dôvodu nám tento typ frontu so zahltením príliš nepomôže. Neumožňuje zvýhodnenie určitých paketov, a teda nezabezpečuje tak ochranu proti aplikáciám zahlcujúcim sieť. Ďalšia nevýhoda nastáva vo chvíli, keď je front plný. Ďalšie prichádzajúce pakety do frontu sú jednoducho zahadzované.

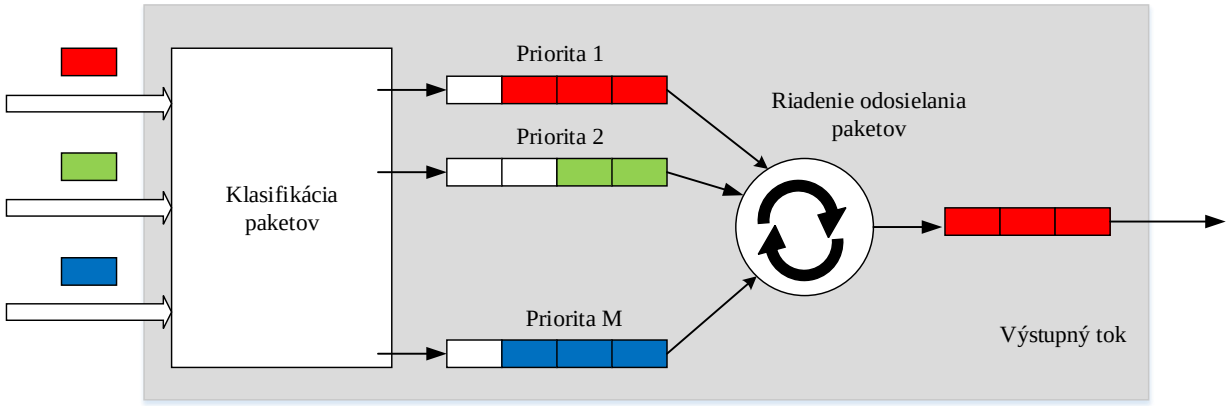


Obr. 3.3.: FIFO front

3.3.2 PQ (*Priority Queuing*)

Jedná sa o typ frontu s prioritnou obsluhou, ktorá zároveň zachováva jednoduchosť frontu typu FIFO. Jednotlivým frontom prideľuje priority ako možné vidieť na nasledujúcom obrázku 3.4. Na vstupe je prichádzajúci paket zaradený podľa stanovených priorít do jedného z front, napr. Priorita1, Priorita2 až Priorita M alebo Vysoká, Stredná, Nízka. Tie pakety, ktoré nemajú pridelenú prioritu, sa automaticky v tomto konkrétnom prípade zaradia do frontu priority „Stredná“. Vždy sa najprv odosielajú pakety z frontu s najvyššou prioritou, až potom prichádzajú postupne na rad fronty s nižšou prioritou. Preto už na prvý pohľad môžeme vidieť, že hlavnou nevýhodou

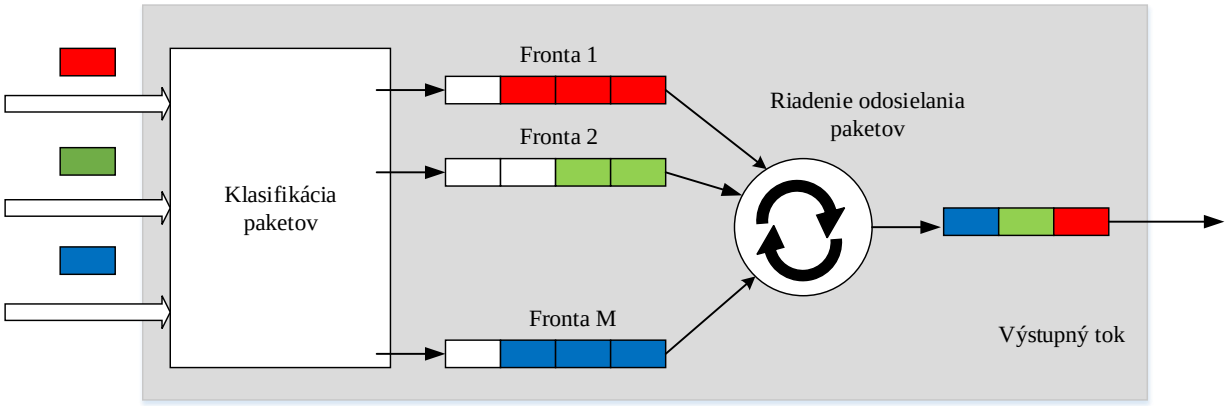
prioritného radenia je, že pakety vo frontoch s niššou prioritou môžu ostať uviaznuté. Pokým sa na ne nedostane rad, pakety z ich frontu nie sú odoslané. Tieto pakety sa vďaka veľkému oneskoreniu javia ako stratené. Pomocou prioritného radenia je možné realizovať urýchlené odosielanie (EF), ktoré je popísané v časti 3.1.3.



Obr. 3.4.:PQ front

3.3.3 FQ (*Fair Queuing*)

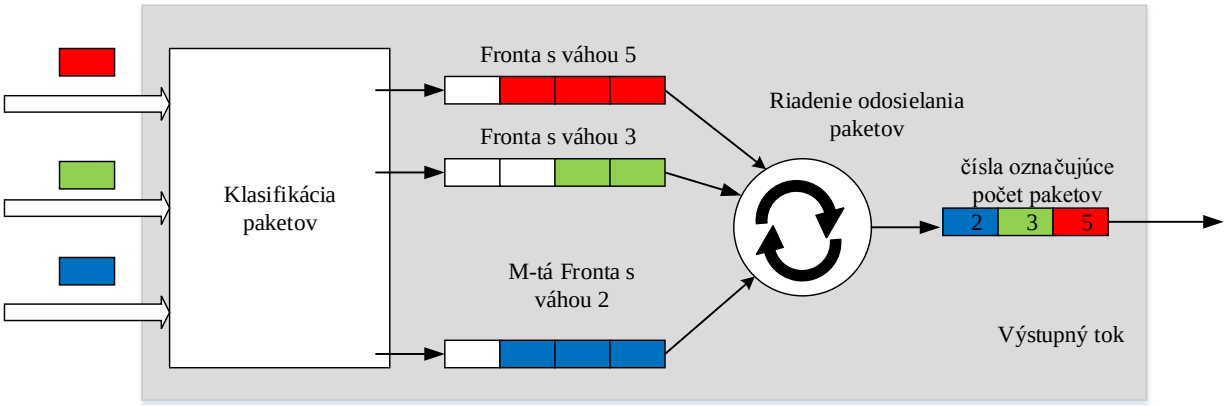
Ako bolo už spomenuté, PQ má určité nevýhody, ktoré metóda FQ vylepšuje v podobe poskytnutia prostriedkov na rovnomerné obsluhovanie jednotlivých frontov práve v čase zahltenia. Prichádzajúce pakety sú zaraďované do N frontov. Každému takémuto frontu je pridelená určitá časť celkovej šírky prenosového pásma výstupného portu. Fronty sú následne cyklicky obsluhované. Nevýhoda tejto metódy oproti PQ je absencia možnosti uprednostniť pakety niektorého frontu, z čoho vyplýva, že FQ nie je moc vhodná pre prenosy citlivé na kolísanie oneskorenia. Avšak dovoľuje zabezpečiť pre vybranú prevádzku minimálnu priepustnosť na linke.



Obr. 3.5.:FQ front

3.3.4 WFQ (*Weighted Fair Queuing*)

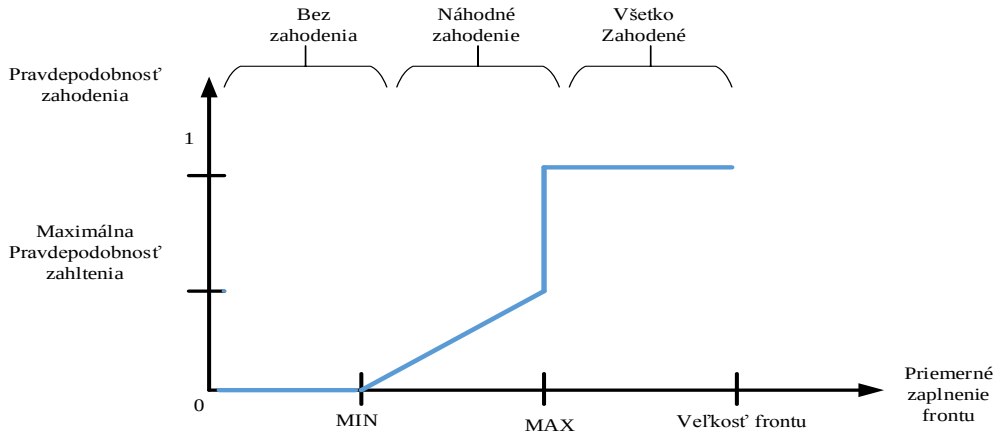
Už zo samotného názvu vieme povedať, že rozhodujúcim faktorom pri tomto mechanizme bude váha paketu. To znamená, že pakety s vyššou prioritou budú v prípade zahltenia zahodené s menšou pravdepodobnosťou. Pre prevádzku s paketmi s vyššou prioritou vyhradzuje algoritmus väčšiu šírku pásma, čím zabezpečuje rýchlejšiu obsluhu dátového toku aj v prípade zahltenia siete. Pre jednoduchšie vysvetlenie môžeme povedať, že sa jedná o istú obdobu FQ. Každý front má pridelenú váhu napr. 5, 3 a 2 paketov alebo 50%, 30% a 20%. Na základe váh jednotlivých frontov sa dané fronty vyprázdňujú. V tomto prípade by sa napr. z prvého frontu odoslalo 5 paketov, potom z druhého 3 a z posledného 2 pakety.



Obr. 3.6.:WFQ front

3.3.5 RED (*Random Early Detection*)

Ide o aktívnu, preventívnu metódu včasnej detekcie s náhodným zahadzovaním ešte pred zahltením frontu. Front je pod neustálym dohľadom a v závislosti od jeho zaplnenia je uplatňované náhodné zahadzovanie paketov. Tento výber zahadzovania sa deje podľa interne stanovených kritérií, ktoré môžu byť zvolené rôzne, napr. podľa obr. 3.7. Od určitej zvolenej hranice začne proces náhodného zahadzovania paketov. Pri nie veľkom zaplnení frontu je pravdepodobnosť zahodenia paketu malá. So stúpajúcim percentom zaplnenia frontu stúpa aj pravdepodobnosť zahodenia paketu. Ak nastane prípad, že obsah frontu je väčší ako zvolené maximum, všetky prichádzajúce pakety sú jednoducho zahadené [11].



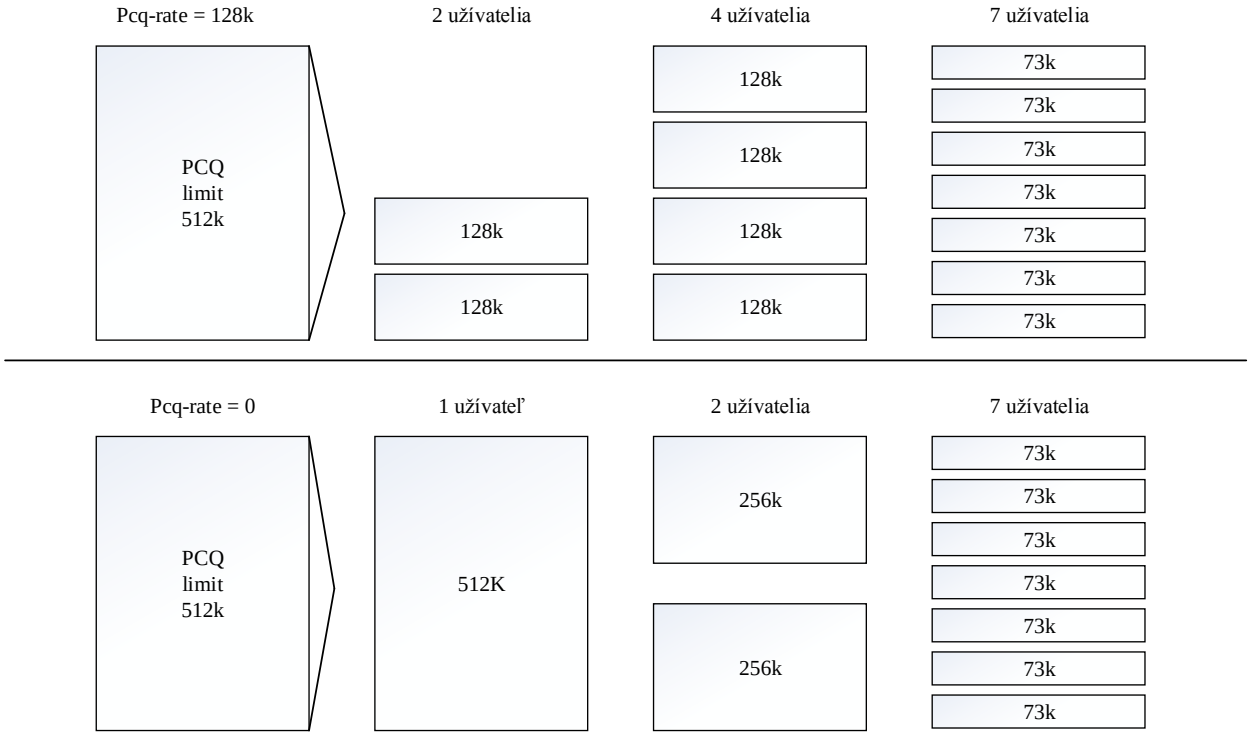
Obr. 3.7.: Graf pravdepodobnosti zahodenia paketu podľa naplnenia frontu (RED)

3.3.6 SFQ (*Stochastic Fairness Queuing*)

Táto metóda zaisťuje manažment frontu pomocou hešovacieho algoritmu a algoritmu *round-robin*. Dátový tok môže byť identifikovaný na základe 4 parametrov, a to zdrojovej adresy, cieľovej adresy, zdrojového portu a cieľového portu. Tieto parametre sú potom použité pri SFQ hešovacom algoritme na klasifikáciu paketov do jedného z 1024 možných FIFO frontov. Následne *round-robin* algoritmus začne rozdeľovať dostupnú šírku prenosového pásma do jednotlivých front s tým, že v jednom cykle umožní odoslať 1514 až 32767 Bytov (záleží na nastavení).

3.3.7 PCQ (*Per connection Queuing*)

PCQ je veľmi podobnou metódou k SFQ, no obsahuje ešte ďalšie funckie. Tak isto ako v predošlom prípade SFQ, aj tu je možné použiť na indetifikáciu dátového toku 4 parametre. Na nasledujúcom obr. 3.8. je možné vidieť rozdelenie jednotlivých dátových tokov pri použití *pcq-rate* (paramater určujúci maximálnu veľkosť pre každý dátový tok) a *pcq-limit* (veľkosť fronty pre jeden dátový tok). V prvom prípade (hore) môžeme vidieť, že aj keď je dostupných 512kB frontu, jednotlivé dátové toky majú pridelených iba 128 kB, čo je maximálna hodna. Následne sú potom jednotlivé dátové toky rozdelované rovnomerne. V druhom prípade môžeme vidieť, že parameter *pcq-rate* je nastavený na hodnotu 0, a tým pádom sú jednotlivé dátové toky vždy rovnomerne rozdelované. Vo výsledku je PCQ algoritmus veľmi jednoduchý, pretože najprv sa pomocou klasifikátoru oddelia jednotlivé dátové toky, potom sa aplikuje na každý FIFO fronta a obmedzenie, a nakoniec sa aplikuje globálne obmedzenie pre celú FIFO frontu.



Obr. 3.8.: Príklad nastavenia PCQ fronty

4. REALIZÁCIA TOOLBOXU

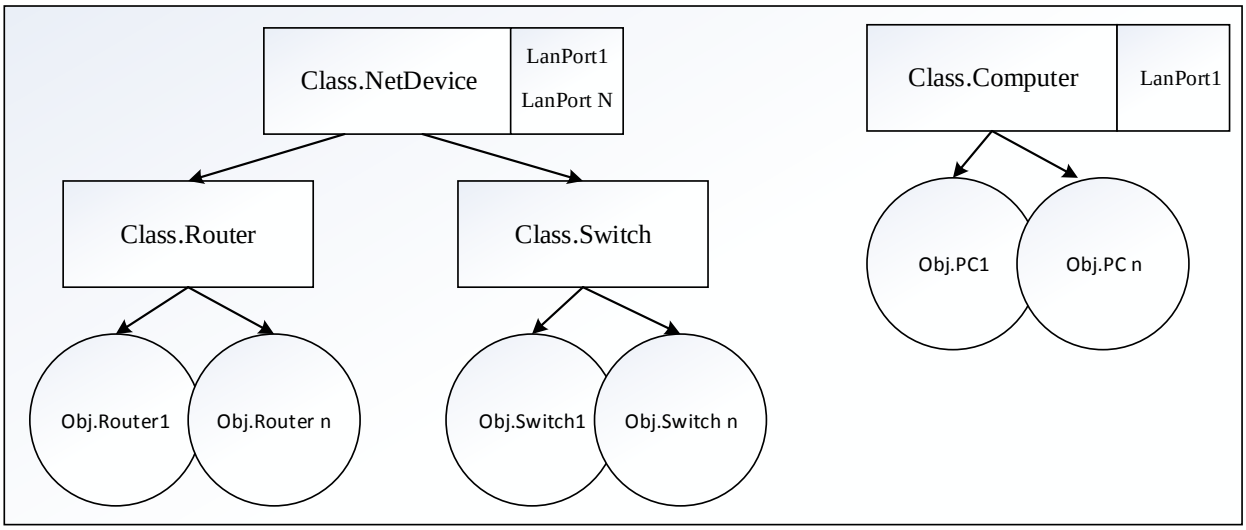
Táto kapitola sa zaoberá realizáciou *toolboxu*, ktorý tvorí praktickú časť práce. V súvislosti s realizáciou budú bližšie spomenuté jednotlivé funkcie a triedy, ktoré boli v *toolboxe* vytvorené. Praktické prevedenie bolo realizované v programovom prostredí Matlab v2014a s využitím techník objektovo orientovaného programovania.

4.1 Triedy a metódy

Trieda tvorí základný pojem klasifikácie na usporiadanie informácie do konkrétnej entity tým spôsobom, že vytvára predpis ako vyrobiť objekt daného typu. Na druhej strane, metódy resp. funkcie poskytujú rozhranie, pomocou ktorého vieme s danými objektami pracovať. V tejto práci boli zadefinované napr. tieto triedy:

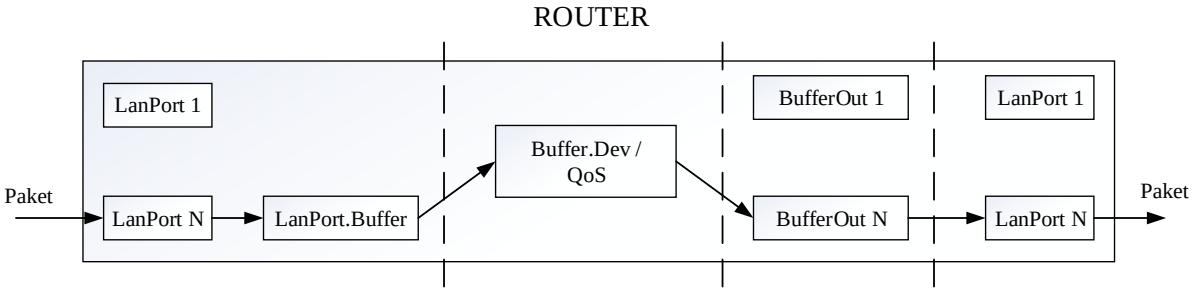
- *Computer* – trieda pre definíciu vlastností objektu počítača.
- *NetDevice* – trieda pre spoločných vlastností prepínača a smerovača.
- *Switch* - trieda pre definíciu vlastností prepínača.
- *Router* - trieda pre definíciu vlastností smerovača.
- *LanPort* – trieda slúžiaca na vytváranie portov pre triedu *Computer* a *NetDevice*
- *Buffer* - trieda slúžiaca na vytváranie zásobníka
- *BufferArray* - trieda slúžiaca na vytváranie pola zásobníkov
- *BufferQOS* – podtrieda triedy *BufferArray* pre QoS zásobníky
- *Results* – slúži na vyhodnotenie výsledkov simulácie (*ResultsFrame*,*Ping*).
- *SimManager* – slúži na menežment všetkých objektov v simulácií

Na obrázku 4.1 môžeme vidieť hierarchiu a použitie tried pre vytváranie jednotlivých objektov. Je jasné, že každý vytvorený objekt alebo teda trieda má niekoľko funkcií, ktoré nám hovoria o tom, aké operárcie môžu uskutočnovať. Na ďalšom obr 4.2 je zobrazený objekt smerovača s popisom



Obr. 4.1.: Hierarchie tried pre vytváranie zariadení

jeho jednotlivých častí. Každý smerovač ma zadefinovaných n portov. Ako pakety prichádzajú do portu smerovača sú postupne ukladané do *LanPort.Bufferu*. V prípade ak nie je prázdny sú jednotlivé pakety poslané do *Buffer*.*Dev/Qos*. V tomto zásobníku sa vykonáva smerovanie paketov alebo operácie súvisiace s *QoS* a pod. Následne sú jednotlivé pakety ukladané do zásobníkov *Buffer*.*Out*, ktorých je rovnaký počet ako portov na smerovači. *Buffer.Out* je zvolený na základe smerovacej tabuľky.



Obr. 4.2.: Objekt Smerovača

Z programového hladiska sú podstatné všetky obsiahnuté funkcie, ale v rámci popisu budú spomenuté iba funkcie, ktoré su popísané nižšie a triedy, ktorým sú priradené.

- Vykreslovanie topológie z *GUI* (nie je trieda ale priamo funkcia).
- Prepájanie zariadení medzi sebou (funkcia *Connect*).

- Smerovanie (funkcia *setSendFrames*).

4.1.1 *NetSim*

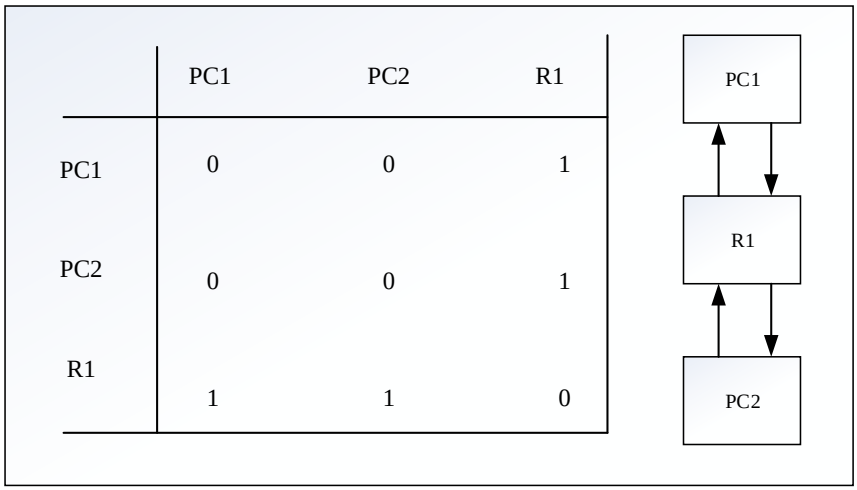
Netsim je funkcia, ktorá slúži na spustenie *GUI* a sú v nej zadefinované jednotlivé operácie, ktoré je možné v *GUI* vykonávať. Medzi ne patrí aj vykreslovanie topológie, ktoré bude bližšie spomenuté. Vykreslovanie je založené, na základe toho, že sa vytvorí matica *cm* (obr.4.3), ktorá je symetrická a má vždy veľkosť podľa toho kolko zariadení sa v sieti nachádza. Po vytvorení tejto matice sa pomocou algoritmu *Equilibrium* vykreslí topológia. Na nasledujúcej strane môžeme vidieť úkážku kódu na vytvorenie topológie.

```
function onShowTopology(~,~)
        cm = zeros(length(devList));
        edgeLength = [];

        for m = 1:length(devList)
            dev_name = devList{m};
            connected_table = M.getByName(dev_name,'connectedTable');
            host_names = connected_table(:,1);

            for n = 1:length(host_names)
                if ~isempty(host_names{n})
                    for p = 1:length(devList)
                        if strcmp(host_names{n},devList{p})
                            cm(m,p) = 1;
                            edgeLength(end+1) = n;
                        end
                    end
                end
            end
        end
        bg = biograph(cm, ...
            devList(:), ...
            'EdgeType', 'curved', ...
            'LayoutType', 'Equilibrium', ...
            'ShowWeights', 'on');
         for m = 1:length(bg.edges)
             bg.edges(m).Weight = edgeLength(m);
         end

        view(bg)
    end
```



Obr. 4.3.: Príklad cm matice

4.1.2 *SimManager*

Ako už bolo spomenuté, táto trieda slúži na menežment všetkých objektov v sieti, ako aj ich funkcií. Jednou z nich je aj konektivita zariadení. Na nižšie uvedom príklade môžeme vidieť, aký spôsobom je táto funkcia zrealizovaná.

```
function Connect(obj,dev_name,port_id)
            % get device Obj
            [dev_type,ind] = obj.getVarName(dev_name);
            dev = obj.(matlab.lang.makeValidName(dev_type)){ind};
            host_name       =   dev.connectedTable{port_id,1};
            port_id_host    =   str2double(dev.connectedTable{port_id,2});
            length          =   str2double(dev.connectedTable{port_id,3});
            type            =   str2double(dev.connectedTable{port_id,4});

            % get host Obj
            [dev_type,ind] = obj.getVarName(host_name);
            host = obj.(matlab.lang.makeValidName(dev_type)){ind};
            dev.Connect(port_id,host,port_id_host,length,type);

            % update / connect host
            host.setConnectedTable(port_id_host, [], {dev_name port_id length
type 1});
            host.Connect(port_id_host,dev,port_id,length,type);
        end
```

4.1.3 *Router*

Táto trieda slúžina vytváranie objektov smerovačov. Ako možno vidieť z obr. 4.1, trieda *Router* je podtriedou triedy *NetDevice*, z ktorej dedí niekoľko vlastností. V rámci tejto triedy bude spomenutá funkcia umožňujúca smerovanie paketov v sieti. Ako prvé sa spraví kontrola stavu

```
function setSendFrames(obj)
          if obj.bufferDev.Status
              frames = obj.bufferDev.popAll;
              for m = 1:length(frames)
                  % get port to send from next packet
                  frame = frames(m);
                  % search route table for port_id
                  if ismember(frame.ipHeader(2,1:3),obj.Ip(:,1:3))
                      dest_ip = frame.ipHeader(2,:);
                      routeTable = obj.routeTableIn;
                  else
                      dest_ip = frame.ipHeader(2,1:3);
                      routeTable = obj.routeTableOut;
                  end

                  for n = 1:length(routeTable)
                      if ismember(routeTable{n}{2},dest_ip)  % get port_id
                          port_id = routeTable{n}{1};
                          break;
                      end
                  end

                  new_dest_mac = [];
                  for n = 1:length(obj.arpTable)
                      if ismember(obj.arpTable{n}{2},dest_ip)
                          new_dest_mac = obj.arpTable{n}{1};
                          break;
                      end
                  end
                      if isempty(new_dest_mac)
                      port_id = 1;
                      new_dest_mac = obj.defaultRoute;
                  end

                  frame.macHeader = [obj.Mac(port_id,:);new_dest_mac];
                  frame.qosHeader
                  obj.bufferOut.Push(port_id,frame);
              end

              for port_id = 1:obj.nPorts
                  if obj.bufferOut.getBuffStatus(port_id)
                      obj.setSendTime(port_id);
                  end
              end
          end
      end
```

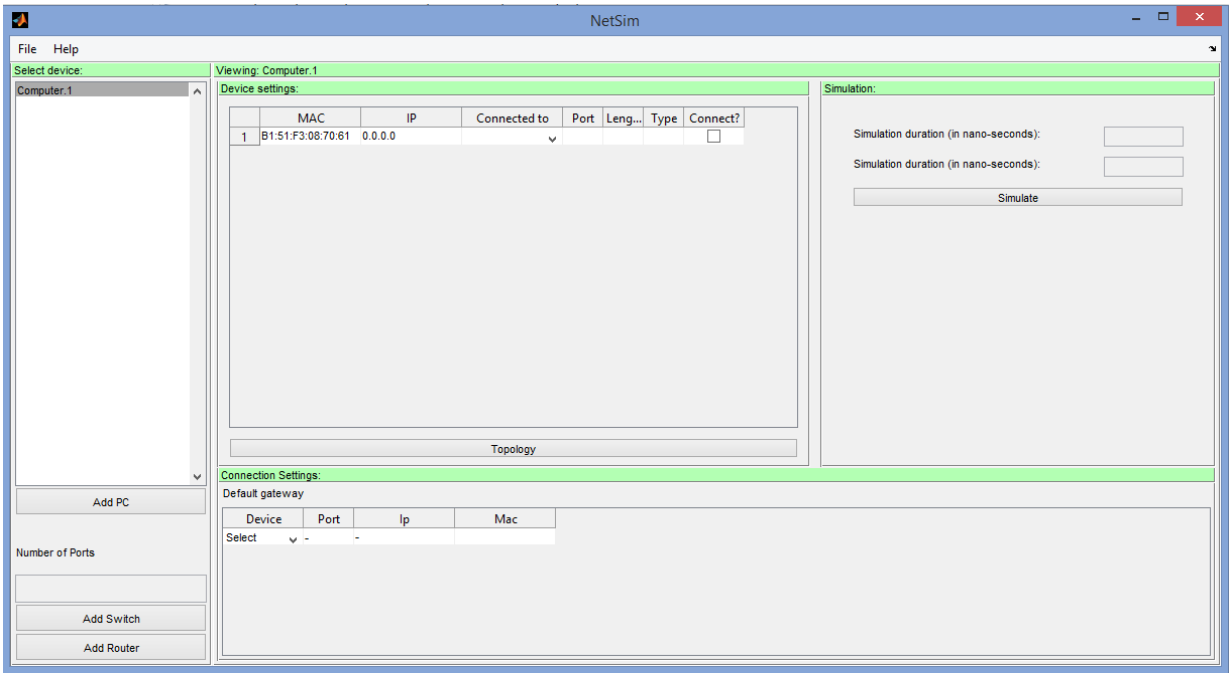
zásobníka *bufferDev*. Ak je plný, vyberie sa určený počet paketov. Toľko, koľko môže v jednom kroku vstúpiť, toľko maximálne vystúpi. Počet je určený počtom portov zariadenia. Ďalej zistíme, či prichádzajúci paket vstupuje do siete v rámci smerovača alebo či je smerovaný do inej siete. V smerovacej tauľke sa nájde port, ktorým bude daný paket odoslaný. Pomocou ARP tabuľky sa zistí, akú má použiť fyzickú adresu. Ak nenájde záznam o adrese v tabuľke, tak priradí fyzickú adresu východzej cesty. Na konci odošle paket na lanport do zásobníka a nastaví čas odoslania, v prípade že nečaká na odoslanie iný paket.

5. GRAFICKÉ ROZHRANIE

Návrh grafického uživateľského rozhrania bol realizovaný pomocou *GUI Layout Toolboxu* [12]. Tento toolbox sa od východzieho Matlab *GUI* odlišuje v tom, že je celý realizovaný v textovom editore. V tejto kapitole budú popísané jednotlivé časti grafického uživateľského rozhrania a práca s ním.

5.1 Popis hlavného panelu a jeho častí

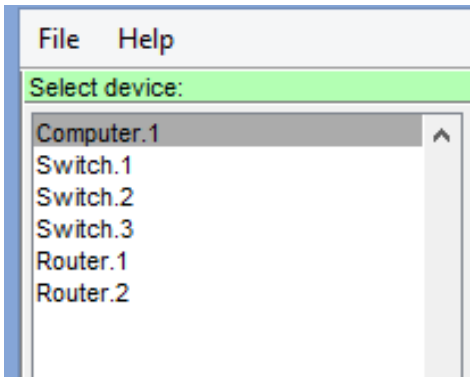
Na obrázku 5.1 je zobrazený hlavný panel menu. Na hlavnom paneli môžeme vidieť niekoľko funkcií, ktoré *GUI* ponúka. Patrí medzi ne pridávanie jednotlivých typov zariadení nachádzajúce sa v ľavej spodnej časti. Ďalej môžeme vidieť v ľavej hornej časti zoznam zariadení, ktoré sú v simulácií obsiahnuté. Panel nachádzajúci sa v hornej a spodnej strednej časti slúži na nastavenie paramametrov zariadení. Ako môžeme vidieť, medzi týmito panelmi je tlačidlo *topology*, ktoré slúži na zobrazenie sieťovej topológie.



Obr. 5.1.: Hlavný panel GUI

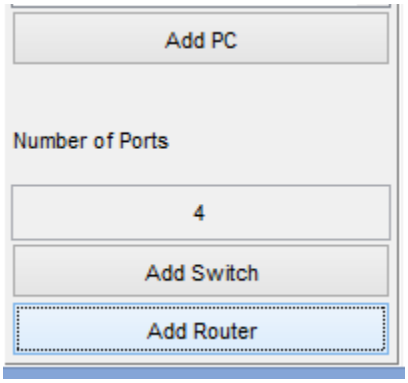
Pravá strana panelu obsahuje možnosti nastavenia simulácie. V rámci grafického uživateľského rozhrania je umožnené uložiť nastavené parametre topológie, a to v ľavej hornej časti pomocou tlačidla *File*, a následne pomocou tlačidla *Save*. Jednotlivé subpanely budú podrobnejšie popísané v nasledujúcej časti.

a) Zoznam zariadení – obsahuje zariadenia, ktoré sú súčasťou testovanej topológie.



Obr. 5.2.: Zoznam zariadení

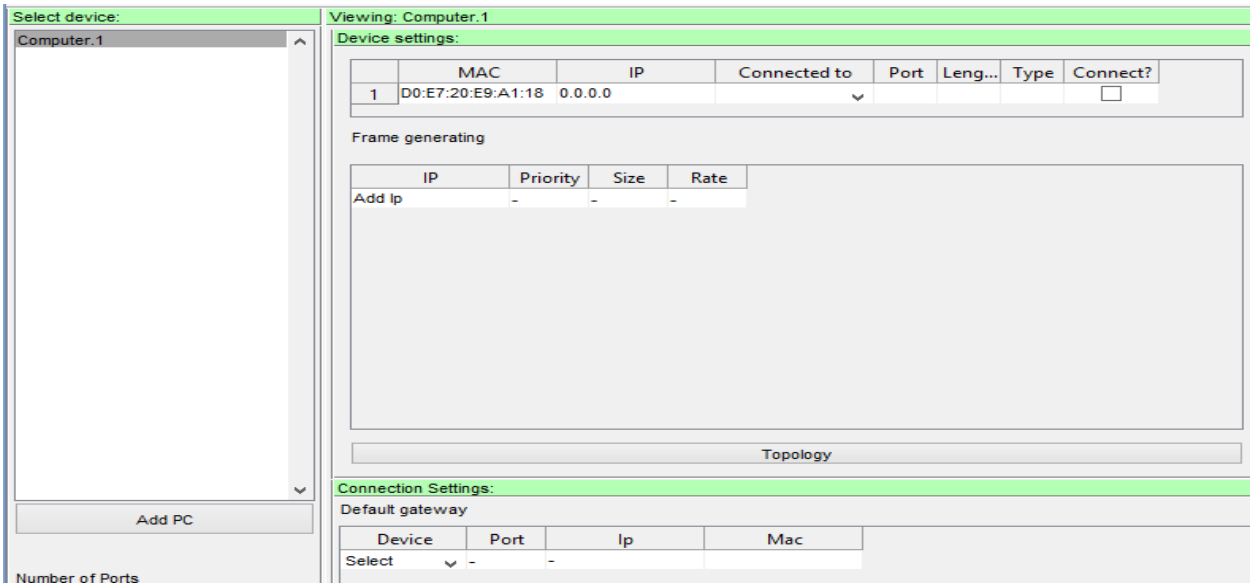
b) Prídávanie zariadení do topológie – umožňuje pridávanie zariadení a špecifikáciu, koľko portov dané zariadenie bude mať.



Obr. 5.3.: Tlačidlá pre pridávanie zariadení

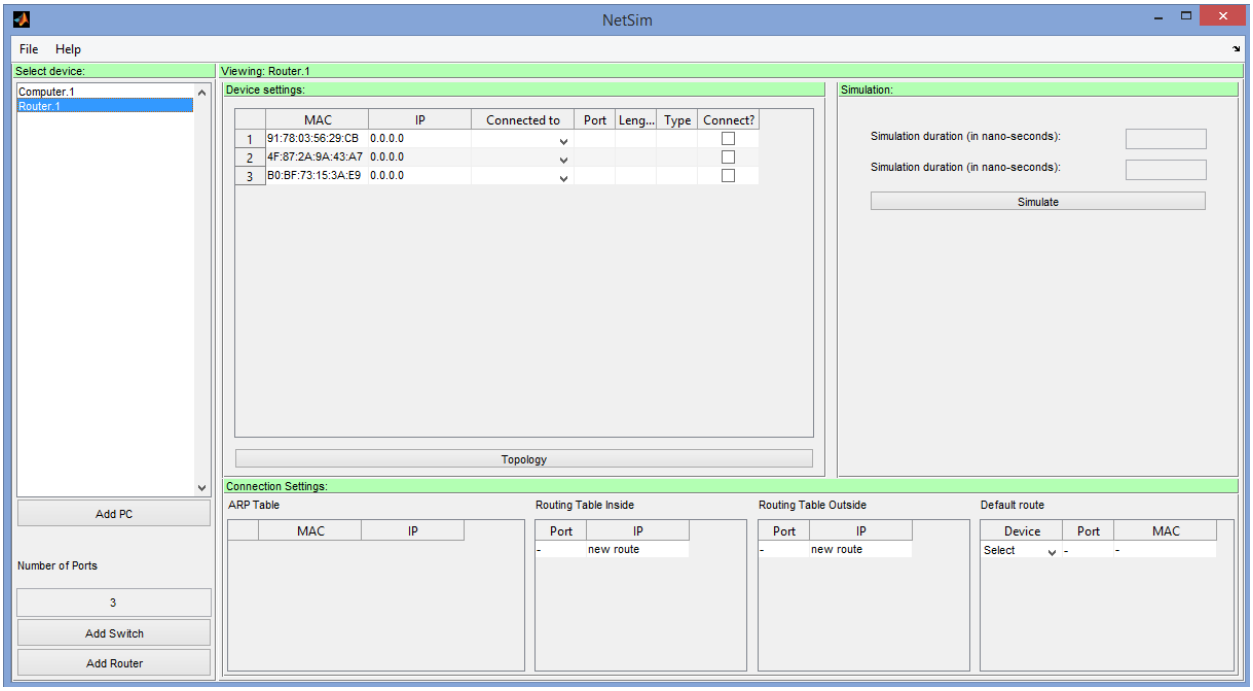
c) Nastavenie parametrov PC – v tomto subpanely môžeme danému koncovému bodu nastaviť niekoľko parametrov, medzi ktoré patrí: logická adresa zariadenia, veľkosť a rýchlosť odosielaných paketov, priorita používanej služby, zariadenia, na ktoré koncový bod pripojíme port prípojného zariadenia, dĺžka kábla, rýchlosť prenosu signálu a nakoniec môžeme pridať

východziu bránu a cieľovú IP kam, na ktorú budú pakety odosielané. Fyzická adresa je zariadeniu vygenerovaná automaticky. Detaily možno vidieť na obr. 1.4.



Obr. 5.4.: Nastavenie parametrov PC

d) Nastavenie parametrov smerovača – podobne ako v predošlom prípade pri nastaveniach PC je aj tu potrebné nastaviť pripojenie jednotlivých portov smerovača na porty opozitných zariadení. Takisto ako pri PC sú fyzické adresy pridelené automaticky jednotlivým portom.

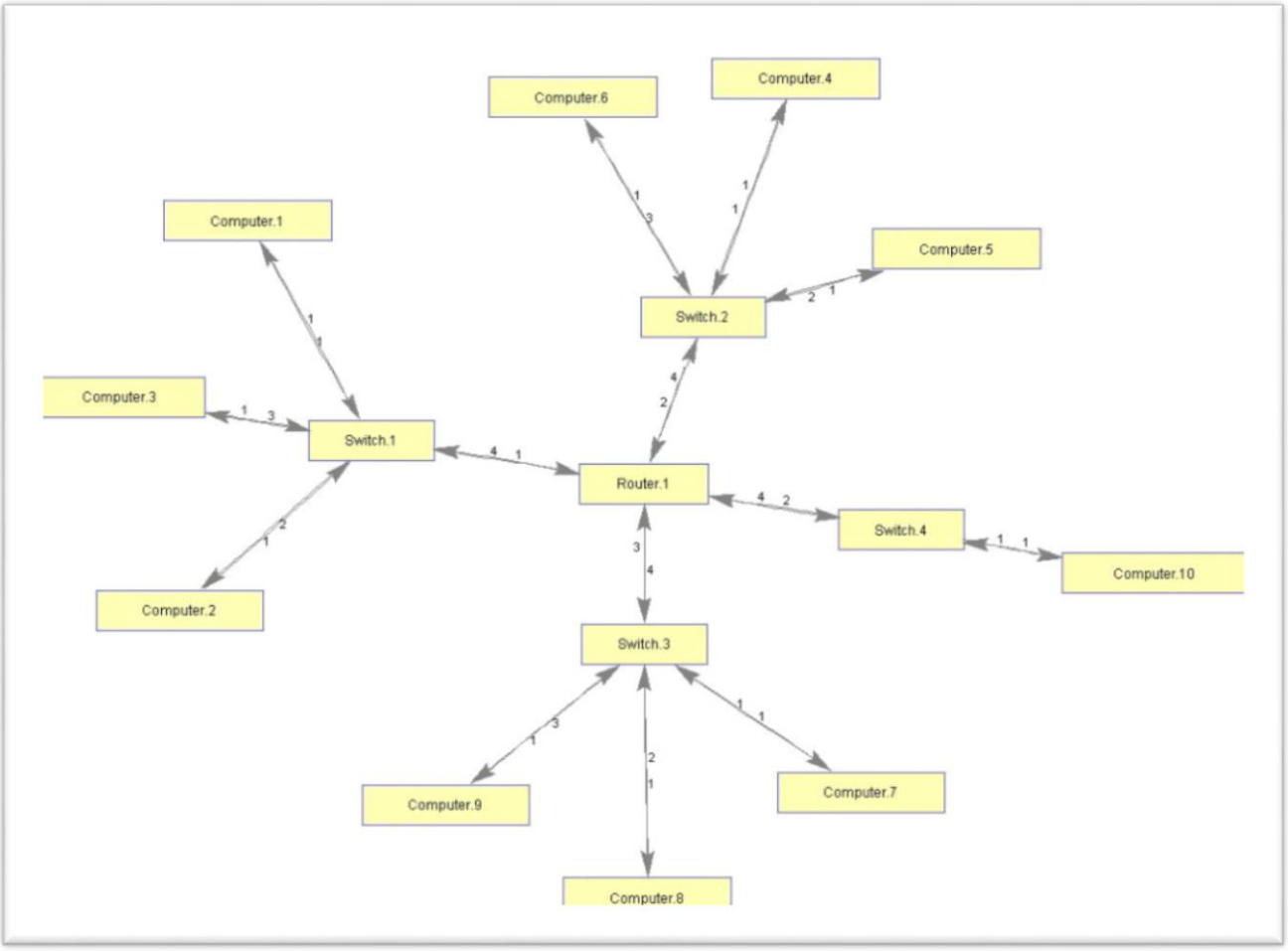


Obr. 5.5.: Nastavenie parametrov smerovača

Logické adresy pre jednotlivé porty je nútne nastaviť manuálne. Ďalej je umožnené nastaviť statické smerovanie, východziu bránu a zobraziť si ARP tabuľku smerovača či jeho smerovaciu tabuľku.

e) Nastavenie parametrov prepínača – v rámci prepínača je možné nastaviť počet portov a zobraziť si MAC tabuľku.

f) Topológia – tlačidlo *topology* slúži na zobrazenie zostavenej topológie. Príklad výstupu tohoto tlačidla možno vidieť na nasledujúcom obrázku obr. 5.6.



Obr. 5.6.: Príklad vytvorenej topológie

g) Nastavenie parametrov simulácie – tu môžeme nastaviť dva parametre simulácie, a to celkovú dĺžku simlácie a krokovanie.

5.2 Spúšťanie simulácie

V rámci práce budú odovzdané dve verzie realizácie *toolboxu*. Jedna z nich je vytvorená pomocou príkazov v príkazvom riadku a druhá umožňuje nastavenie pomocou *GUI*. Pre spustenie simulácie pomocou *GUI* je použitá funkcia *netsim* a pre spustenie *toolboxu* v príkazovom riadku je to funkcia *TestRouting*. Úvodné menu v rámci GUI je zobrazené na obr.5.1. Nastavenie jednotlivých parametrov a pod. bolo bližšie popísné v predchádzajúcej kapitole. Nastavenie simulácie pomocou príkazového riadku je zobrazené na nasledujúcom príklade časti kódu.

```
M.addComputer;  // vytváranie objektov, (X) počet portov
M.addSwitch(7);
M.addRouter(4);

M.Connect(M.Switches{1},2,M.Computers{1},1,20,2e8)
// nastavenie konektivty prepínač, PC.
// Definícia parametrov pre kábel (číslo portu zariadenia, dlžka kábla, rýchlost odosielani)

M.Computers{1}.queueFrame(M.Computers{10}.Ip,ones(randi([1200 1500]),1));
//nastavenie komu bud[ pakety posielan0 PC1 > PC 10
M.Routers{1}.addIp(1,[192 168 1 1]);
// priradenie IP adresy na port smerovača

M.Computers{m}.addIp([192 168 1 10]);
M.Computers{m}.addGate(M.Routers{1}, 1);
// priradenie IP adresy na PC a astavenie východzej brány.

M.Routers{1}.addRoute(1,M.Computers{m}.Ip);
// Statické nastavenie smerovania

sim_step = 1e6;         // nastavenie kroku simulácie
sim_length = 5e9;       // nastavenie dĺžky simulácie v nano sekundách napr. 5sec simulácie
```

5.2.1 Výstupy simulácie

Pomocou triedy *SimManger* je možné na konci každej simulácie nahliadnuť do všetkých zariadení a pozrieť stavy jednotlivých ich parametrov. Medzi ne patria napr. stav v akom sa nachádzajú jednotlivé zásobníky, aké majú zariadenia pridelené fyzické a logické adresy, ako je naplnená smerovacia tabuľka, ARP tabuľka alebo MAC tabuľka a pod.

ZÁVER

Táto práca sa zaoberala možnosťou modelovania simulácie v ethernetových sieťach. Teoretická časť práce pojednáva o technológii *Ethernet* a jej štandardoch a o kvalite služby v týchto sieťach. Bližšie spomenuté boli mechanizmy spojené so zabezpečením kvality pomocou diferencovaných služieb.

Praktickú časť tvorí realizácia *toolboxu*, ktorý umožňuje vytvorenie dynamickej siete. V tejto sieti môžu byť obsiahnuté zariadenia; koncový bod (počítač), prepínač a smerovač. V rámci koncového bodu je možné nastaviť nasledujúce parametre: logická adresa, dĺžka kábla, ktorým je počítač pripojený k opozitnému zariadeniu , rýchlosť prenosu dát v médiu, cieľovú stanicu, na ktorú budú pakety odosielané a rýchlosť odosielania paketov. V rámci prepínača je možné definovať počet portov, ktoré bude mať. Smerovač umožňuje komplexné nastavenie jeho parametrov, a to: logické adresy, ktoré je možné manuálne prideliť jednotlivým portom, nastavenie statického smerovania v sieti a zobrazenie *ARP* tabuľky.

Všetky spomenuté vlastnosti je možné nakonfigurovať pomocou grafického užívateľského rozhrania, ktoré bolo zrealizované pomocou *GUI Layout toolboxu*. Súčasťou grafického užívateľského rozhrania je možnosť zobrazenia nakonfigurovanej topológie siete. Na tejto topológii sú zobrazené zariadenia a jednotlivé spojenia, na ktorých sú označené čísla portov zariadení.

Vytvorený *toolbox* bol realizovaný pomocou techník objektovo orientovaného programovania v programovacom prostredí *Matlab 2014a*. *Matlab* pôvodne nebol vytvorený na prácu s objektovo orientovaným programovaním (OOP). Manipulácia s OOP je miestami zložitá, no napriek tomu sa po krátkom čase javí ako celkom výhodná metóda realizácie.

Kým *toolbox* v počiatkoch realizácie fungoval prostredníctvom príkazového riadku, nevyskytovali sa žiadne komplikácie. Avšak veľkou nevýhodou je náročné a pracné nastavenie simulácie, kedy je potrebný veľký počet riadkov kódu slúžiaci na nastavenie zariadení a topológie ako takej.

Realizácia grafického uživateľského rozhrania začala spôsobovať väčšie komplikácie, čo zabralo veľa času. Prvotná myšlienka vytvorenia grafického rozhrania bola riešená pomocou *GUIDE*, ktorý je implementovaný v *Matlabe*. Dochádzalo však k častému vypisovaniu chybových hlásení bez logických oddôvodnení, na základe čoho sa presunula práca na grafickom uživateľkom prostredí do *GUI Layout toolboxu*. Tento *toolbox* je odporúčaný na viacerých diskusných *Matlab* fórach ako lepšia varianta a zakladá sa na manuálnom písaní a spravovaní. Vo výsledku malo *GUI* vytvorené prostredníctvom *GUI Layout toolboxu* 650 riadkov. Sú v ňom obsiahnuté funkcie a vlastnosti zariadení, vykreslenie topológie a pod.

Pri implementácií vlastností ako ukladanie navrhnutej topológie, či merania parametrov kvality služby však začalo dochádzať k viacerým chýbám, ktoré sa nepodarili autorovi vyriešiť. Ďalším z problémov bolo naviazanie na „*backend*“. *GUI* používa všetko ako reťazce a ťažko pracuje s objektami, čo bolo príčinou potreby konvertovania všetkých vstupov. Často bolo nutná konvertácia zo „*cell array*“ na reťazec, a potom na čísla. Tento fakt spôsobil nutnosť zmien v „*backende*“, čo však predstavuje problém, nakoľko GUI by malo „*backend*“ využívať a nie určovať jeho štruktúru. Posledný veľký problém predstavovalo ukladania simulácie. Matlab nedokáže ukladať „*event Listenery*“, čím je znehodnotený celý objekt *SimManager*. Aj keď je jeho štruktúra zachovaná, všetky väzby sú zrušené. Napísať kód, ktorý by tieto väzby obnovil, je takmer nemožné, nakoľko by sa skákalo od reťazca k reťazcu a parsovali by sa sem a tam názvy objektov, čo by aplikácia, ktorá slúži na výpočty, robiť nemala. Toto viedlo k postupnej strate funkcionality, až sa nakoniec stal „nepoužitelným“.

Na základe týchto faktov bola presunutá realizácia naspäť do príkazového riadku. Je možné porovnať oba kódy a pozrieť sa na ich komplexnosť, napr. funkcia *connect* v *SimManager*, prípadne funkcie na parsovanie reťazcov ako názvov objektov aleob metód, napr. *getByName*, *setByName*. Z týchto dôvodov boli v rámci tejto práce odovzdané obe verzie *toolboxu*, aby bolo čitateľovi umožnené overiť si, aké rozdiely su v kóde a vo funkcionalite.

Zoznam použitej literatúry

[1] FAIRHURST, G.: *Ethernet*, 2009. Dostupné na: http://www.erg.abdn.ac.uk/users/gorry/eg3567/lan-pages/enet.html

[2] POOLE, I.: *Ethernet Tutorials, Telecommunications and Networks*. Dostupné na: http://www.radio-electronics.com/info/telecommunications_networks/ethernet/ethernet-ieee-802-3-standards.php

[3] VEGESNA, S.: *IP Quality of Service [online]*, Indianapolis, USA: Cisco Press, 2001. ISBN 1-57870-116-3.

[4] CISCO SYSTEMS: *Voice Quality, Understanding Delay in Packet Networks [online]*. Dostupné na: http://www.cisco.com/en/US/tech/tk652/tk698/technologies_white_paper09186a00800a8993.shtml

[5] *Integrated Services in the Internet Architecture*, an Owerview, RFC 1633.

[6] *An Architecture for Differentiated Services*, RFC 2475.

[7] CISCO SYSTEMS: *DiffServ The Scalable End-to-End QoS Model [online]*. Dostupné na: http://www.cisco.com/en/US/technologies/tk543/tk766/technologies_white_paper0918

[8] CISCO SYSTEMS: *Implementing Quality of Service Policies with DSCP [online]*. Dostupné na: http://www.cisco.com/en/US/tech/tk543/tk757/technologies_tech_note09186a00800949f2.shtml

[9] CISCO SYSTEMS: *Classification Overview[online]*. Dostupné na: http://www.cisco.com/en/US/tech/tk543/tk757/technologies_tech_note09186a00800949f2.shtml

[10] CISCO SYSTEMS: *Congestion Management Overview [online]*. Dostupné na: http://www.cisco.com/en/US/docs/ios/qos/configuration/guide/congstion_mgmt_oview_ps6350_tsd_Products_Configuration_Guide_Chapter.html

[11] NORMIS (admin): *MikroTik documentation [online]*. Dostupné na: http://wiki.mikrotik.com/wiki/Manual:Queue#PCQ

[12] TORDOFF B., SAMPSON D. GUI Layout toolbox *[online]*. Dostupné na: http://www.mathworks.com/matlabcentral/fileexchange/47982-gui-layout-toolbox

[13] The MathWorks, Inc. Matlab Documentation [online]. Dostupné na: http://www.mathworks.com/help/matlab/

[14] ATTAWAY, S. MATLAB: a practical introduction to programming and problem solving. 2nd ed. Waltham: Butterworth-Heinemann, c2012, xx, 518 p. ISBN 978-0-12-385081-2.

Prílohy

Priložené CD obsahuje dve verzie *ethernetového toolboxu* a DP v elektronickej verzii. Zložka s názvom *GUI toolbox* obsahuje verziu *toolboxu* realizovanú v *GUI* a zložka *toolbox* obsahuje bez implementácie *GUI.*