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

NÁVRH NOVÝCH LABORATORNÍCH ÚLOH V SÍŤOVÉM SIMULÁTORU OMNET++

NEW LABORATORY EXERCISES IN OMNET++ NETWORK SIMULATOR

DIPLOMOVÁ PRÁCE
MASTER'S THESIS

AUTOR PRÁCE — Bc. Martin Vlašín
AUTHOR

VEDOUCÍ PRÁCE — Ing. Lukáš Langhammer
SUPERVISOR

BRNO 2016

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ

Diplomová práce

magisterský navazující studijní obor **Telekomunikační a informační technika**
Ústav telekomunikací

Student: Bc. Martin Vlašín ***ID:*** 134660
Ročník: 2 ***Akademický rok:*** 2015/16

NÁZEV TÉMATU:

Návrh nových laboratorních úloh v síťovém simulátoru OMNET++

POKYNY PRO VYPRACOVÁNÍ:

Náplní diplomové práce je návrh čtyř nových laboratorních úloh v simulačním prostředí OMNeT++. Prostudujte možnosti simulačního prostředí OMNeT++ a na základě možností tohoto prostředí vytvořte laboratorní úlohy včetně kompletních návodů vhodných pro studenty zahrnující i doplňující úkoly a kontrolní otázky. Při návrhu úloh se zaměřte na okruhy: protokol MPLS, BGP,OSPF, srovnání IPv4 a IPv6, porovnání transportních protokolů TCP, UDP, SCTP, srovnání technologií Ethernet a WLAN, zajištění kvality služeb v síti, či rozbor aplikačního protokolu (jako ICMPv6, DNS, popřípadě další). Časová náročnost každé úlohy musí být přibližně dvě hodiny.

DOPORUČENÁ LITERATURA:

[1] FOROUZAN, Behrouz A. TCP/IP protocolsuite. 4th ed. Boston: McGraw-HillHigherEducation, 2010, xxxv, 979 s. ISBN 978-0-07-337604-2.

[2] OMNeT++ User Manual , 2015. Dostupné online < https://omnetpp.org/documentation>

[3] JEŘÁBEK, J. Komunikační technologie. Brno: Vysoké učení technické v Brně, 2013. s. 1-172.

Termín zadání: 1.2.2016 ***Termín odevzdání***: 25.5.2016

Vedoucí práce: Ing. Lukáš Langhammer.
Konzultant diplomové práce:

doc. Ing. Jiří Mišurec, CSc., *předseda oborové rady*

UPOZORNĚNÍ:

Abstrakt

Diplomová práce se zabývá návrhem čtyř laboratorních úloh s využitím simulačního nástroje OMNeT++ a rozšířením Inet Framework. Témata navržených úloh jsou zaměřena na směrovací protokoly, zajištění kvality síťových služeb, technologie použité v lokálních počítačových sítích a srovnání vlastností aktivních síťových prvků. První úloha obsahuje srovnání vlastností a směrovacích technik protokolů RIPv2 a OSPFv2, včetně vlivu rozdělení OSPF sítě na menší oblasti. Druhá úloha se zaobírá technikami pro řízení kvality služeb, konkrétně frontou FIFO, PQ a WFQ. Cílem je srovnání dosažených hodnot parametrů definujících kvalitu síťových služeb pro jednotlivé techniky. Třetí úloha srovnává technologie Ethernet a WLAN, parametry přenosu. Čtvrtá úloha nabízí srovnání vlastností síťových prvků pro tvorbu počítačových sítí, rozbočovač, přepínač a směrovač. V práci je také popsána potřebná teorie síťových technologií a programových prostředků používaných k provedení simulací.

Klíčová slova

OMNeT++, Inet Framework, QoS, DiffServ, FIFO, PQ, WFQ, RIPv2, OSPFv2, Ethernet, 802.3, WLAN, 802,11, rozbočovač, přepínač, směrovač, laboratorní úlohy, modelování sítí, simulování sítí.

Abstract

Diploma thesis deals with creating four laboratory tasks using simulation tool OMNeT++ and Inet Framework extension. Subject matter of these designed tasks is focused on routing protocols, providing quality of service, computer networking technologies used in local area networks and comparison of active networking devices. First task contains the comparison of behaviour and routing techniques between protocols RIPv2 and OSPFv2, including the effect of dividing OSPF network into smaller areas. Second task is focused on various techniques for controlling quality of service, namely FIFO queue, PQ and WFQ. Main goal is the comparison of results, that define quality of network services. Third task compares networking technologies, Ethernet and WLAN, parameters of transmission. Fourth task offers comparison of networking devices, hub, switch and router. This paper also contains required theory of network technologies and software tools used for executing simulations.

Keywords

OMNeT++, Inet Framework, QoS, DiffServ, FIFO, PQ, WFQ, RIPv2, OSPFv2, Ethernet, 802.3, WLAN, 802.11, hub, switch, router, laboratory tasks, network modeling, network simulating.

VLAŠÍN, M. *Návrh nových laboratorních úloh v síťovém simulátoru OMNeT++*. Brno: Vysoké učení technické v Brně, Fakulta elektrotechniky a komunikačních technologií, 2016. 93 s. Vedoucí diplomové práce Ing. Lukáš Langhammer.

PROHLÁŠENÍ

Prohlašuji, že svou diplomovou práci na téma „Návrh nových laboratorních úloh v síťovém simulátoru OMNeT++“ jsem vypracoval samostatně pod vedením vedoucího diplomové práce a s použitím odborné literatury a dalších informačních zdrojů, které jsou všechny citovány v práci a uvedeny v seznamu literatury na konci práce.

Jako autor uvedené diplomové práce dále prohlašuji, že v souvislosti s vytvořením této diplomové práce jsem neporušil autorská práva třetích osob, zejména jsem nezasáhl nedovoleným způsobem do cizích autorských práv osobnostních a jsem si plně vědom následku 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 c.40/2009 Sb.

Brno ……. …….

(podpis autora)

SIX research centre
sensor, information and communication systems

Faculty of Electrical Engineering
and Communication

Brno University of Technology
Technicka 12, CZ-61600 Brno, Czechia

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

Výzkum popsaný v této diplomové práci byl realizovaný v laboratořích podpořených projektem Centrum senzorických, informačních a komunikačních systémů (SIX); registrační číslo CZ.1.05/2.1.00/03.0072, operačního programu Výzkum a vývoj pro inovace.

MINISTERSTVO ŠKOLSTVÍ, MLÁDEŽE A TĚLOVÝCHOVY

EVROPSKÁ UNIE
EVROPSKÝ FOND PRO REGIONÁLNÍ ROZVOJ
INVESTICE DO VAŠÍ BUDOUCNOSTI

2007–13
OP Výzkum a vývoj pro inovace

Poděkování

Děkuji vedoucímu práce Ing. Lukáši Langhammerovi za velmi užitečnou metodickou pomoc a cenné rady při zpracování diplomové práce.

V Brně dne …….. .....

podpis autora

Obsah

ÚVOD

Cílem této práce je prouzkoumání možností síťového simulátoru OMNeT++ a následná aplikace těchto poznatků při vytváření praktických úloh, jež by vhodným způsobem posloužili studentům k rozšíření, či názornému ověření jejich teoretických znalostí o počítačových sítích a v nich používaných technologiích. Praktickým výstupem této práce jsou čtyři kompletní laboratorní úlohy, včetně srozumitelně popsaného postupu k vytvoření, nastavení a následné analýze výsledků. Tak aby studenti mohli využít co možná nejvíce času simulováním a testováním parametrů sítě, namísto zdlouhavého navrhování sítě a vytváření programového kódu. Začlenění těchto úloh spadá do předmětové stavby bakalářského studia a dle toho byly vybrána témata a jejich náročnost.

Téma první laboratorní úlohy je zaměřeno na srovnání vlastností směrovacích protokolů uvnitř autonomních sítí, konkrétně protokoly RIP (Router Information Protocol) a OSPF (Open Shortest Path First). Druhá úloha se zabývá metodami zajištění kvality služeb a vliv jejich použití v závislosti s nastavenými parametry k docílení požadované kvality přenosu dat. Třetí úloha srovnává vlastnosti a parametry síťových technologií. Těmi jsou kabelové sítě dle standardu IEEE 802.3 - Ethernet a bezdrátové sítě dle standardu IEEE 802.11 – WLAN. Čtvrtá úloha porovnává funkce a vlastnosti aktivních síťových prvků pro tvorbu sítí, jmenovitě se jedná o rozbočovač, přepínač a směrovač.

Pro návrh projektů je využito grafické prostředí programu OMNeT++ s využitím nástavby INET Framework, která zahrnuje popis důležitých síťových prvků a technologií. Jedná se o tzv „open source“ kód a používání těchto programů je pro nekomerční účely zcela bezplatné. K zobrazení výsledků jsou využity integrované nástroje obsažené v programu OMNeT++.

V kapitole 1 jsou popsány jednotlivé části síťového simulátoru OMNeT++ a nástroje použité k vytváření projektů, provádění simulací a zobrazení výsledků.

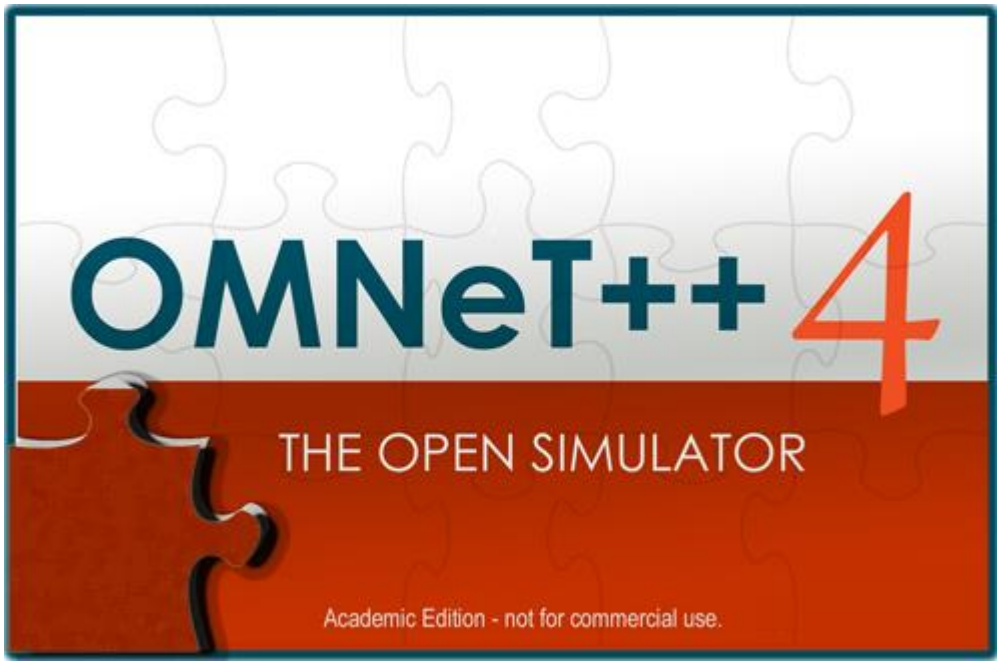
V následujících čtyřech kapitolách je teoreticky popsána problematika témat, z nichž jsou sestaveny laboratorní úlohy. Kapitola 2 popisuje problematiku zajištění kvality služeb, kapitola 3 obsahuje teorii směrovacích protokolů, kapitola 4 popisuje síťové technologie pro tvorbu lokálních sítí a v kapitole 5 jsou uvedeny charakteristiky aktivních síťových prvků.

Kapitola 6 obsahuje praktickou část, stručný popis čtyř vytvořených laboratorních úloh. V této kapitole je pouze nastíněno zadání úloh včetně topologie simulované sítě a vypracování, které bude po studentech požadováno. Kompletní návody jsou k nalezení v příloze.

1 SÍŤOVÝ SIMULÁTOR OMNET++

OMNeT++ je simulační nástroj zaměřený na síťové technologie, jeho základ je postaven na platformě Eclipse, kterou rozšiřuje dalšími funkcemi. Programové prostředí je založeno na objektově orientovaném programování v jazyce C++. Jádro programu je distribuováno jako otevřený kód a lze tak nalézt nová rozšíření či vylepšení, ovšem tento způsob vývoje nevylučuje výskyt chybných implementací. Pro tuto práci je použita v současnosti aktuální verze OMNeT++ 4.6 a dále rozšíření s názvem Inet Framework ve verzi 3.1, které obsahuje implementaci většiny z nejdůležitějších protokolů linkové, síťové a transportní vrstvy, včetně podpory bezdrátových a mobilních technologií. Rozšíření Inet je podrobněji popsáno v kapitole 1.4.

Model OMNeT++ se skládá z modulů, které mezi sebou komunikují předáváním zpráv. Jednoduché moduly jsou popsány v jazyce C++ a využívají knihovnu tříd. Spojováním modulů vznikají moduly složené, jejichž hierarchie není nijak omezena. Zprávy mezi nimi jsou typicky přenášeny jejich naprogramovanými vstupními a výstupními branami. Vlastnosti těchto bran jsou závislé na programové definici, mohou fungovat jako virtuální spojení pro účel převodu informací do požadovaných částí složených modulů, či mohou simulovat spoje z reálných podmínek přiřazováním typických parametrů, jako je chybovost, zpoždění nebo přenosová rychlost. [1]

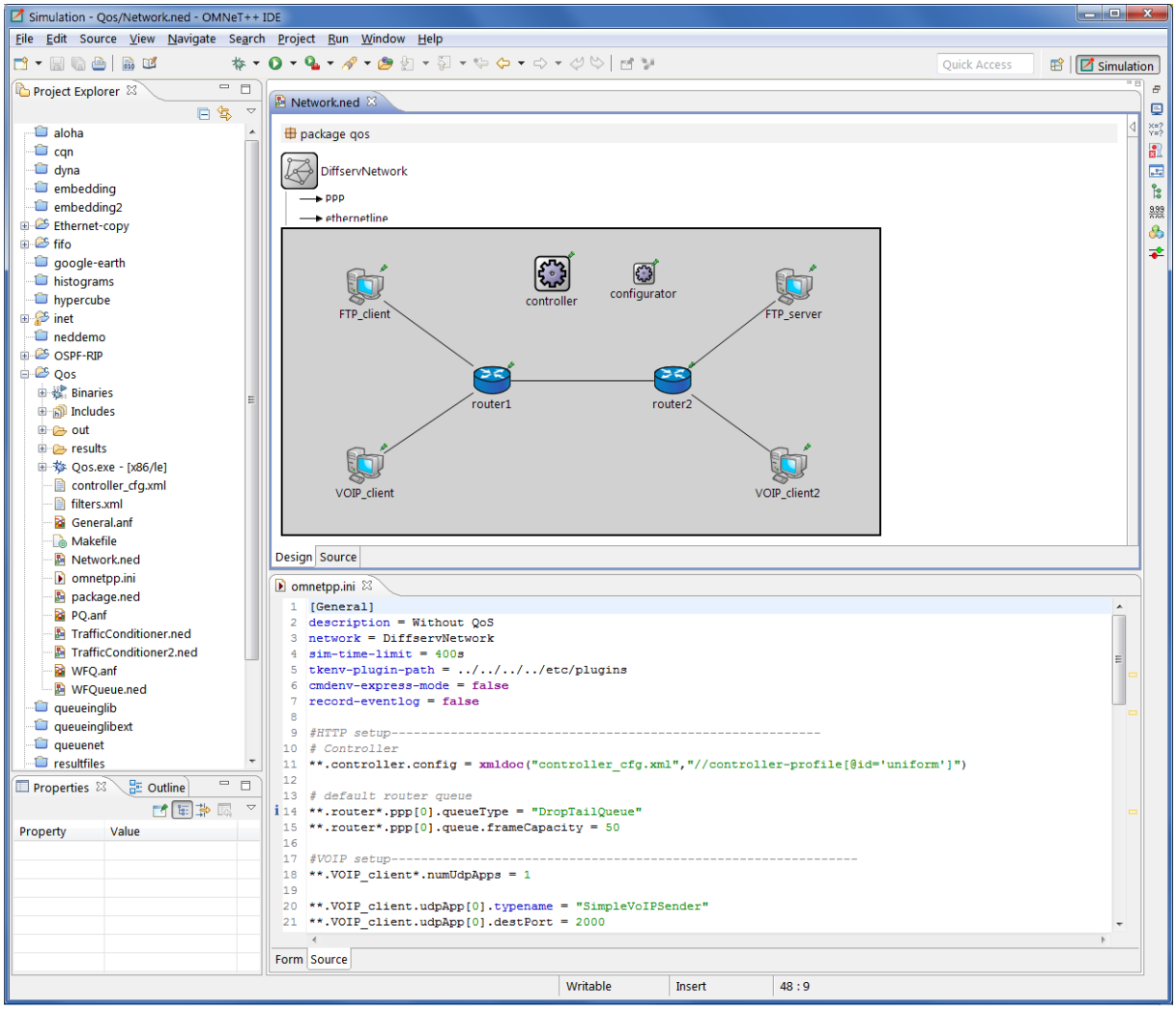


Obr. 1: Úvodní obrazovka OMNeT++ 4.6

1.1 Vývojové prostředí

Vývojové prostředí programu OMNeT++ je založeno na platformě Eclipse a nabízí pokročilé možností editování a formátování kódu přizpůsobené potřebám konfiguračních souborů, které slouží k návrhu topologií a popisu funkčních částí pro potřeby simulací a následné analýzy výsledků.

Pracovní plocha nabízí bohaté možnosti zobrazení potřebných částí a doplňků, Project Explorer nabízí stromovou struktura pracovního adresáře, ve kterém jsou všechny aktuální projekty, editovací část pro zobrazení a psaní kódu a informační část s konzolí, debuggerem pro zobrazování chyb a problému. Ukázku pracovního prostředí lze vidět na Obr. 2.

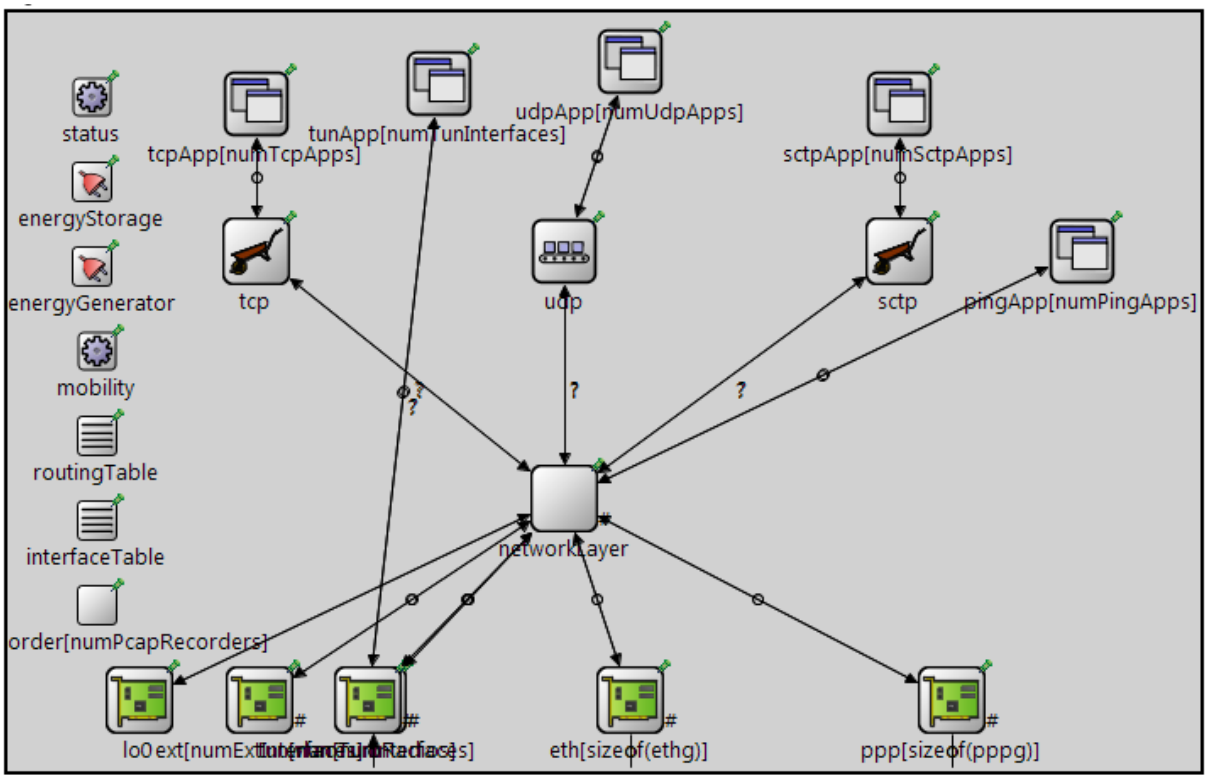


Obr. 2: Pracovní prostředí programu OMNeT++ IDE

Forma zápisu se liší podle typu konfiguračního modelu, základním prvkem pro tvorbu prvků jsou tzv. moduly napsané v jazyce C++, které definují základní funkčnost. Spojováním těchto bloků vznikají ekvivalenty reálných technologií a síťových zařízení. Ty jsou popsány ve specifickém jazyce NED (Network Description) a pro jeho tvorbu lze využít NED editor, který nabízí dva módy.

NED editor

Grafický mód, ve kterém lze propojovat moduly z knihovny prvků předdefinovanými kanály. Tento mód názorně prezentuje hierarchii modulů a jejich vzájemných propojení, ze kterých se vytváří síťová zařízení a technologie. Na Obr. 3 je zobrazena vnitřní struktura prvku StandardHost, který představuje implementaci osobního počítače v jazyce NED. Prvek je tvořen jednotlivými moduly, které definují například vstupní a výstupní rozhraní, aplikace, směrovací tabulky a další.



Obr. 3: Vnitřní struktura prvku StandardHost

Textový mód umožňuje uživateli přímý přístup ke zdrojovému kódu NED prvků, kde jsou zapsány implementace všech modulů, vstupní a výstupní brány, jejich propojení a také definice všech parametrů, které lze nastavovat. Syntaxe vychází z jazyka C++. [2]

Pomocí NED editoru se dají vytvářet topologie sítí, ale není vhodné v nich definovat konkrétní parametry, jelikož NED soubory jsou statické a jejich zadání je tak pevně dané a nelze ho měnit. Pro definici parametrů a síťových aplikací je vhodné použít INI File Editor, který slouží k vytváření konfiguračních souborů INI popisujících chování simulované sítě i simulace samotné.

INI File Editor

Umožňuje konfiguraci simulačního modelu, nastavení parametrů simulace, filtrování požadovaných výsledků, vytváření aplikací a datových přenosů na síťových zařízeních, adresování a spoustu dalších možností. Parametry mohou být nastavovány jednoznačně nebo rozsahem, případně měnící se v čase. Každý konfigurační soubor obsahuje povinnou sekci General, parametry v ní zapsané jsou společné pro všechny části simulace. Je možné také vytvářet scénáře a k nim přiřadit jejich vlastní parametry.

Podkladem konfiguračního souboru je topologie sítě vytvořená v souboru NED, parametry pak lze nastavovat podle částí, jež jsou obsaženy v topologii. Například aplikace TCP (Transmission Control Protocol) je možné definovat pro jakýkoliv prvek, který má v NED kódu definován modul tcpApp. Na obrázku Obr. 4 je zobrazena názorná ukázka kódu konfiguračního souboru.

```
1  [General]
2  description = Without QoS
3  network = DiffservNetwork
4  sim-time-limit = 400s
5  tkenv-plugin-path = ../../../../etc/plugins
6  cmdenv-express-mode = false
7  record-eventlog = false
8
9  #HTTP setup-----------------------------------------------------------
10 # Controller
11 **.controller.config = xmldoc("controller_cfg.xml","//controller-profile[@id='uniform']")
12
13 # default router queue
14 **.router*.ppp[0].queueType = "DropTailQueue"
15 **.router*.ppp[0].queue.frameCapacity = 50
16
17 #VOIP setup-----------------------------------------------------------------
18 **.VOIP_client*.numUdpApps = 1
19
20 **.VOIP_client.udpApp[0].typename = "SimpleVoIPSender"
21 **.VOIP_client.udpApp[0].destPort = 2000
22 **.VOIP_client.udpApp[*].destAddress = "VOIP_client2"
23 **.VOIP_client.udpApp[*].startTime = 0s
24 **.VOIP_client.udpApp[*].stopTime = 70s
25
26 **.VOIP_client2.udpApp[0].typename = "SimpleVoIPReceiver"
27 **.VOIP_client2.udpApp[0].localPort = 2000
28
```

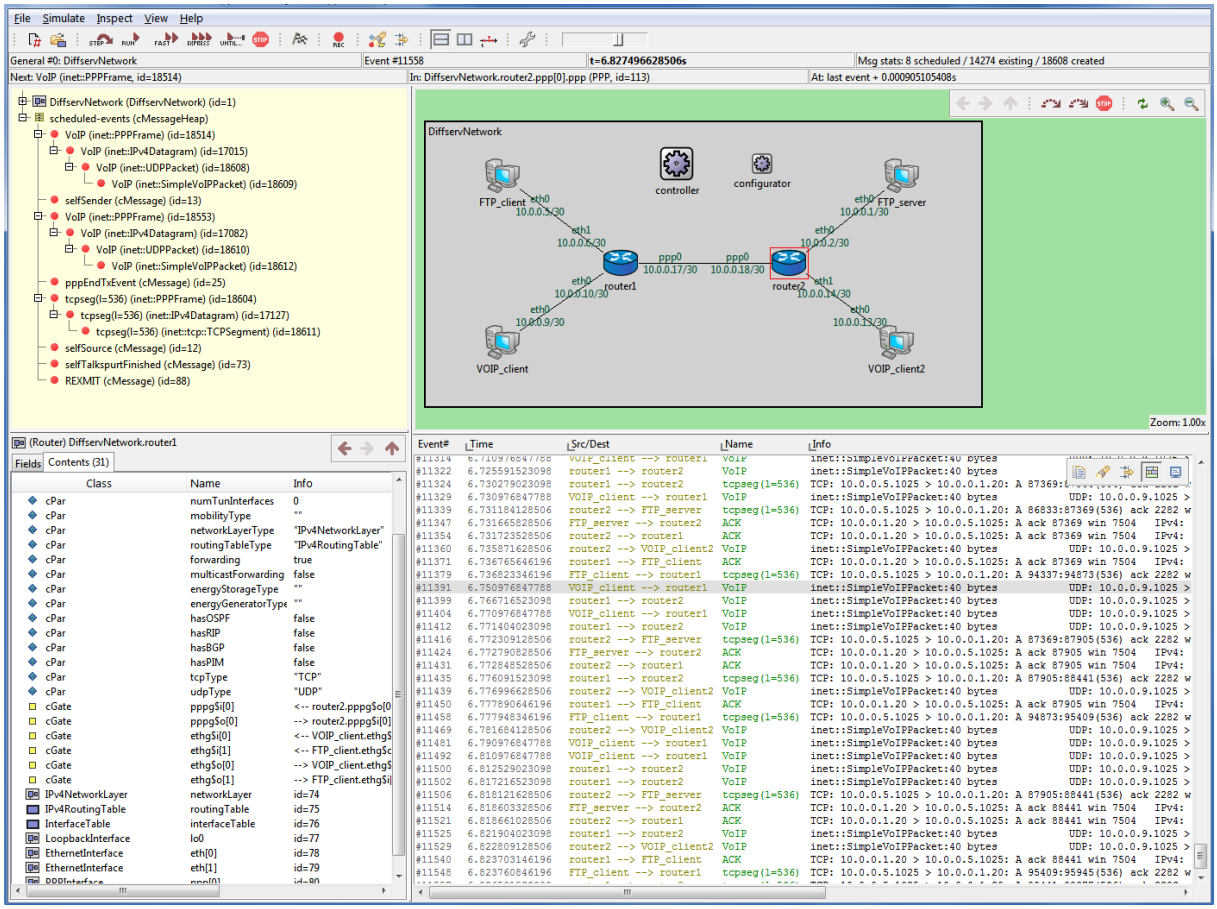
Obr. 4: Ukázka konfiguračního souboru

Vytvořením těchto dvou souborů, NED a INI, je projekt zkompletován a připraven k simulaci. Nejprve je nutné projekt zkompilovat, aby se sestavily a vytvořili všechny potřebné části a poté lze zahájit simulaci spuštěním konfiguračního souboru. Program OMNeT++ poskytuje vlastní simulační prostředí, které je popsáno v následující kapitole.

1.2 Simulační prostředí

Síťový simulátor OMNeT++ nabízí integrované nástroje pro simulování vytvořených projektů. Spuštěním vytvořených scénářů v konfiguračním souboru se spustí nástroj Tkenv, který nabízí spoustu možností pro sledování událostí simulace. Příklad zobrazení simulačního okna je zachycen na Obr. 5.

V grafické části okna lze sledovat průběh simulace v animovaném režimu, který vyobrazuje všechny přenášené zprávy přímo v síťové topologii, případně lze vhodným filtrem zobrazit pouze vyžadovaný typ zpráv. Po vybrání síťového prvku z topologie lze v okně s obsahem, v levém dolním rohu, zkoumat všechny části, jež prvek obsahuje. Tedy aktivní rozhraní, směrovací tabulky a podobně. V části s textovým výstupem událostí, v pravém dolním rohu, jsou vypsány všechny zprávy přenášené v rámci topologie. Pro jejich popis slouží zdroj a cíl zprávy, typ segmentu, čas a pořadí v rámci simulace nebo informace o obsahu. Po otevření konkrétního rámce jej lze zkoumat do detailů, jako velikost zprávy, velikost hlaviček, typ zapouzdření a další. V levém horním rohu je výpis naplánovaných událostí, které se dají zobrazit i na časové ose.

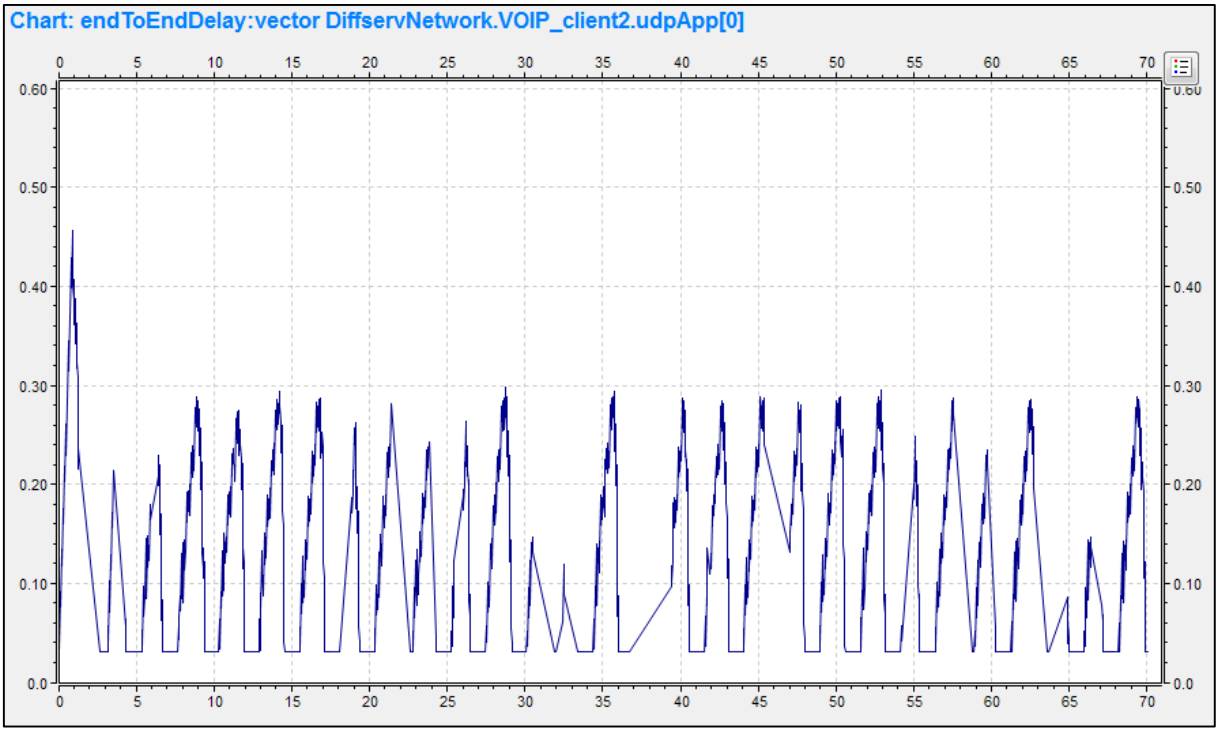


Obr. 5: Simulační prostředí OMNeT++/Tkenv

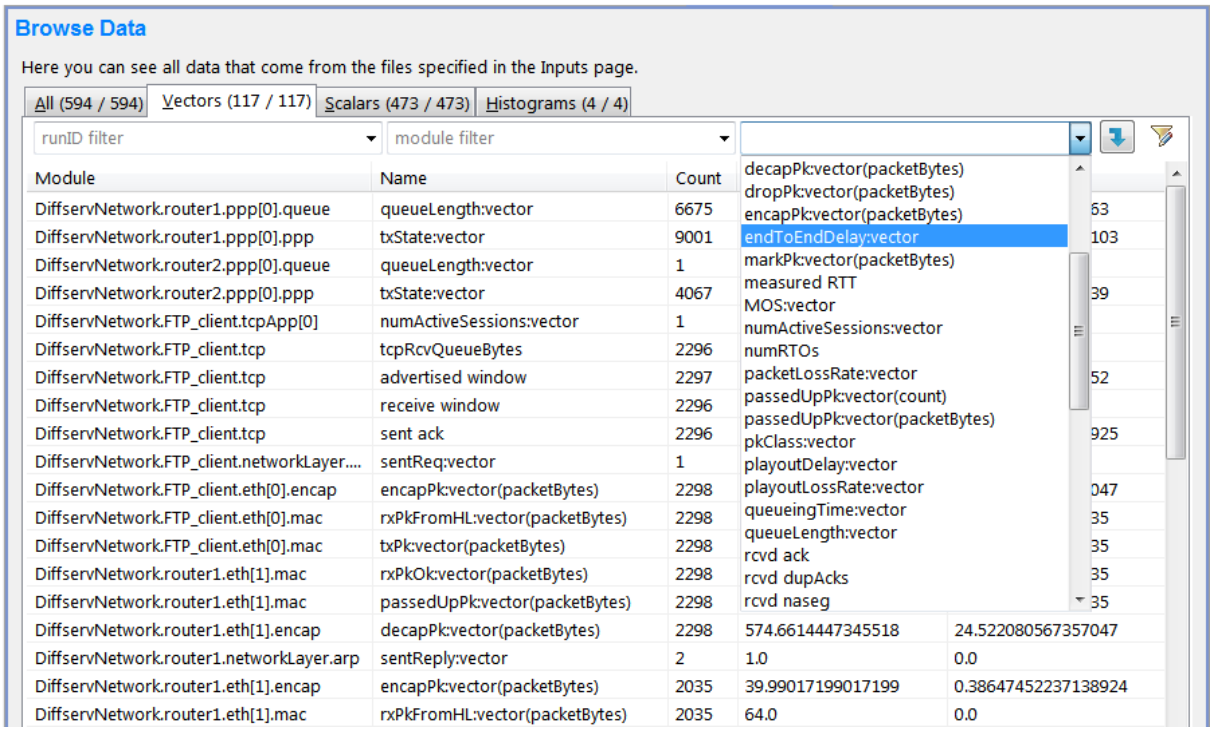
Simulaci je možné provádět skokově, libovolnou rychlostí nebo krokovat po jednotlivých událostech. Simulace je dokončena dosažením časového limitu nastaveného v konfiguračním souboru nebo při úspěšném provedení všech naplánovaných událostí. Po dokončení simulace se vytvoří soubory s výsledky, ty jsou popsány v následující kapitole.

1.3 Analýza výsledků

OMNeT++ obsahuje integrovanou podporu pro analýzu výsledků provedených simulací. Výsledky se ukládají jako výstupní vektor hodnot v čase simulace nebo jako skalární hodnoty, volitelně i jako histogram. Pokud není manuálně zakázáno, provádí se nahrávání hodnot na všech modulech a kanálech. Otevřením souboru s příponou .anf se dostaneme do prohlížeče všech naměřených hodnot, k vyhledání požadované statistiky vybereme patřičnou záložku (Vektors/Scalars), kde je pro zjednodušení hledání možnost využití dvou filtrovacích polí. Z kontextové nabídky nebo zapsáním filtrovací podmínky lze vybrat statistiky pro požadovaný modul (např. koncová stanice, rozhraní ethernet, typ protokolu) nebo konkrétní statistiku (např. zpoždění, délka fronty, odeslané pakety). Na řádku příslušného modulu jsou vypsány naměřené parametry, počítadlo, střední hodnota, odchylka, minimální, maximální hodnota a další). Okno s výpisem naměřených statistik je ilustrováno na Obr. 7. Požadovanou statistiku je také možné vykreslit do grafu, na ose X je vždy čas a na ose Y je parametr vybrané statistiky. Ukázka grafického průběhu zpoždění VOIP (Voice over Internet Protocol) hovoru je zachycena na Obr. 6.



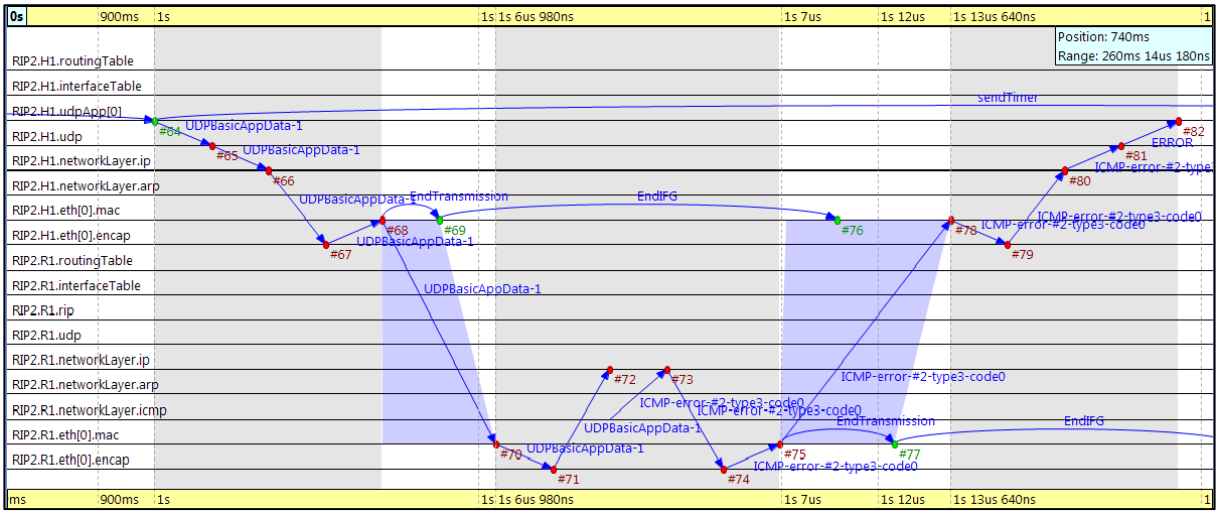
Obr. 6: Ukázka grafického zobrazení průběhu zpoždění



Obr. 7: Nabídka pro analýzu výsledků

Výsledky ve skalární podobě nabízí jedinou výslednou hodnotu, která je sumarizací daného parametru za celou dobu simulace (např. zahozené pakety, přijaté pakety a další).

Volitelnou možností výstupu je tzv. EventLog, který lze povolit zadáním příkazu „record-eventlog = true“ do konfiguračního souboru. Výstupem je výpis předávání zpráv mezi moduly v síti na časové ose, ukázka je vykreslena na Obr. 8.



Obr. 8: Ukázka souboru EventLog

1.4 Rozšíření Inet Framework

Rozšíření Inet je zdarma ke stažení a instalaci lze provést automaticky ve vývojovém prostředí program OMNeT++. Nebo manuálně, stažením aktuální verze zdrojových souborů z webových stránek http://inet.omnetpp.org, následným rozbalením do složky nastavené jako pracovní adresář.

Inet Framework rozšiřuje prostředí OMNeT++ o implementace protokolů (IPv4 - Internet Protocol version 4, IPv6 - Internet Protocol version 6, TCP - Transmission Control Protocol, SCTP - Stream Control Transmission Protocol, UDP - User Datagram Protocol) a řady aplikačních modelů. Podporuje automatickou konfiguraci adresování pro statické směrování a směrovací protokoly (RIP OSPF, BGP - Border Gateway Protocol). Dále nabízí podporu standardů spojové vrstvy (PPP - Point-to-Point Protocol, Ethernet, standard bezdrátových sítí 802.11), multiprotokolové přepojování podle návěští MPLS (Multiprotocol Label Switching), mechanizmy pro řízené přepojování paketů a simulování bezdrátových a mobilních komunikačních sítí. [3]

Vychází z principů zavedených programem OMNeT++, jednotlivé moduly komunikují předáváním zpráv. Pro sestavení modelů reprezentujících protokoly a technologie využívá jednoduché moduly z databáze OMNeT++.

Moduly jsou organizovány do hierarchické struktury, která zhruba odpovídá vrstvám referenčního modelu ISO/OSI (*inet.applications, inet.transportlayer, inet.networklayer, inet.linklayer, inet.physicallayer).* Další části jsou *inet.power, inet.environment, and inet.node*.

2 ZAJIŠTĚNÍ KVALITY SLUŽEB

V počátcích komunikačních sítí bylo poskytováno malé množství síťových služeb, a tak nedocházelo k výraznému vytížení komunikačních linek. Pro všechny typy služeb byly zavedeny jednotné požadavky a úroveň kvality poskytované služby nebyla nijak řešena. Z dnešního pohledu se takovýto způsob přenosu nazývá „Best-Effort“.

V současné době je zavedeno velké množství síťových služeb s odlišnými požadavky na parametry přenosu. Hrozí riziko vysokého vytížení komunikačních linek a bez řízení úrovně kvality by bylo velmi omezeno využití určitých služeb, zejména těch pracujících v reálném čase, jejichž hlavním požadavkem je nízké časové zpoždění a jeho kolísání (jitter). Naopak pro většinu datových služeb je hlavním požadavkem co nejnižší ztrátovost či chybovost paketů a propustnost. Proto bylo nutné zavedení mechanizmů rozlišování typů služeb podle jejich typických požadavků a zajištění odlišných úrovní kvality služeb.

Pro zajištění kvality služeb je základním požadavkem rozlišení toku dat do tříd, podle typu dat zdrojových aplikací. Poté je pro každou třídu definováno odlišné zacházení, mechanizmus pro zajištění kvality služeb je označován pojmem QoS – Quality of Service. Ten také zajišťuje dohled nad síťovým provozem, aktivní správu front jednotlivých tříd, plánuje odesílání paketů a upravuje provoz pro lepší využití dostupné kapacity komunikační linky. [4]

Třídění datových jednotek tvoří dva základní úkony: [4]

- klasifikace paketů – podle informací obsažených v hlavičce
- značkování paketů – přidělením identifikátoru označí příslušnost dat do vybrané třídy

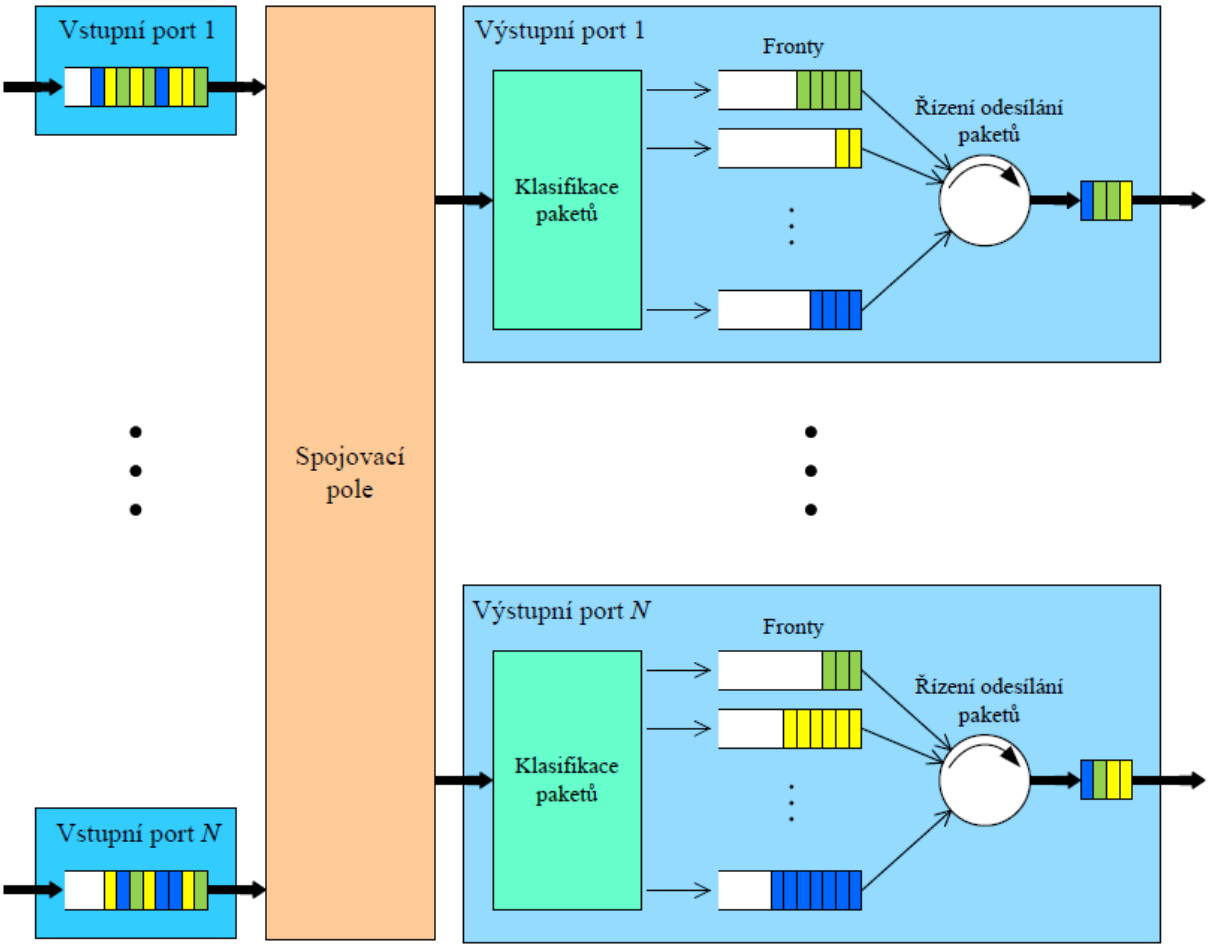
Značkování paketů je obvykle provedeno nastavením hodnoty příslušného pole v hlavičce IP datagramu. Pro značení paketů jsou využita osmibitová pole v hlavičce, ToS (Type of Service) a DSCP (DiffServ Code Point). [5]

Do pole ToS musí nastavit požadovaný charakter přenosu vysílací stanice. Skládá se ze tří bitů určujících prioritu při zpracování paketu, dokáže tak rozlišit sedm úrovní priority, dvě nejvyšší jsou vyhrazeny pro použití při řízení sítě (např. směrovací protokoly), následující tři bity slouží k nastavení požadavků pro přenos (zpoždění, propustnost a spolehlivost přenosu). Zbylé dva bity jsou nevyužity. [6] Tento způsob značkování má velmi omezené možnosti a v současné době už se nepoužívá, byl nahrazen architekturou DiffServ (Differentiated services).

DiffServ využívá stejné pole v hlavičce IP datagramu, ale mění způsob zápisu. Využívá šest bitů k označení třídy, zbylé dva bity jsou označeny jako „dočasně nevyužité“ [4]. Přiřazením k jednotlivým třídám se rozlišuje pouze relativní priorita, jelikož způsob zacházení s třídami je určena konfigurací směrovače.

2.1 Řízené odesílání paketů

Podle označení paketů v hlavičce jsou datové toky přiřazovány do příslušných front, s odlišným způsobem odesílání. Pro definované třídy musí mechanizmus řízení zajistit požadované síťové prostředky. Plánované odesílání paketů je prováděno zvlášť každým výstupním portem směrovače, názorná ukázka je zobrazen na Obr. 9. Nejprve jsou pakety zařazeny na vstup příslušného portu, přeneseny na odpovídající výstupní port dle údajů ve směrovací tabulce. Každý výstupní port provádí klasifikaci paketů a řadí je do odpovídající fronty, ze které jsou posílány na výstup v pořadí závislém na použité metodě řízeného odesílání, případně nastavení jejich parametrů.



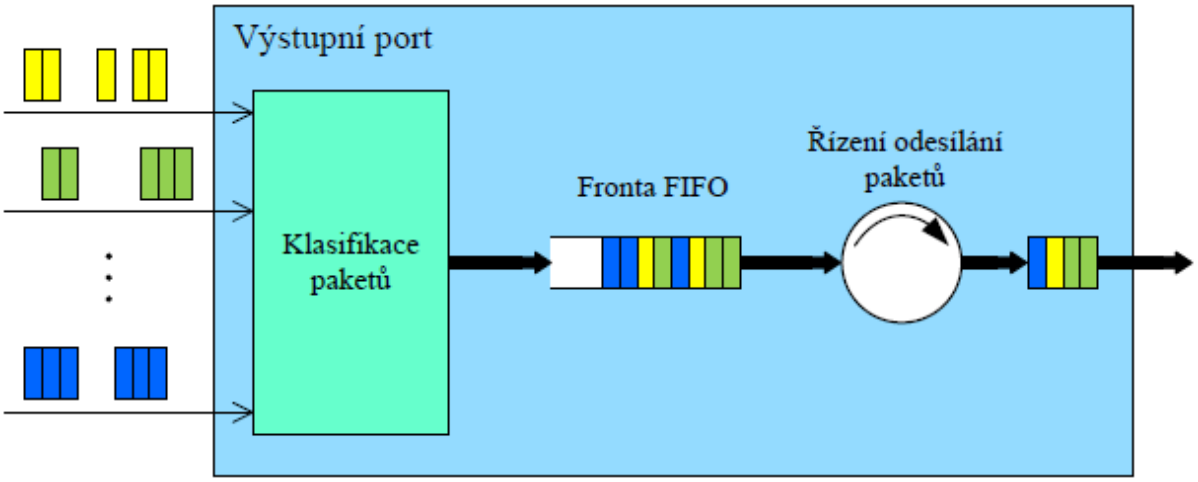
Obr. 9: Řízení odesílání paketů [7]

Existují různé metody pro řízené odesílání paketů [4], mezi nejběžnější patří:

- FIFO (First-In-First-Out)
- PQ (Priority Queueing) - prioritní systém front
- FQ (Fair Queueing) - systém se spravedlivou obsluhou
- WRR (Weighted Round Robin) - systém front s váženou cyklickou obsluhou
- WFQ (Weighted Fair Queuing) - systém front s váženou spravedlivou obsluhou
- CB WFQ (Class-Based Weighted Fair Queuing) - systém front založený na třídách s váženou spravedlivou obsluhou

2.1.1 FIFO

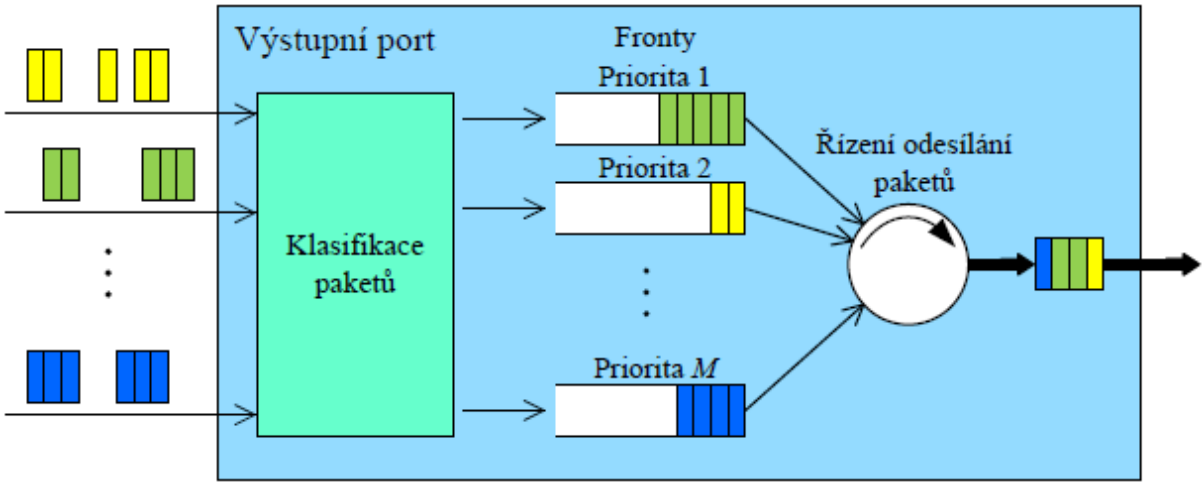
Je základním typem fronty a používá se vždy, když není definován žádný specifický algoritmus pro řízení odesílání. Fronta FIFO představuje zásobník pro příchozí pakety, které jsou na výstup posílány ve stejném pořadí, v jakém byly přijaty. Použití je vhodné především pro sítě s garantovaným způsobem přenosu „Best-Effort“, jelikož nedokáže rozlišovat mezi třídami služeb a zajistit tak požadované parametry, výhodou je velmi jednoduchá implementace. Princip fungování fronty lze vidět na Obr. 10. Velikost fronty je teoreticky neomezena, v praktickém řešení je ve většině případů kapacita fronty omezena maximálním počtem paketů a po dosáhnutí této hranice jsou nově příchozí datové jednotky zahazovány.



Obr. 10: Schéma fronty FIFO [7]

2.1.2 PQ

Jednoduchý prioritní systém front obsluhy, umožňuje diferencování mezi jednotlivými třídami datového provozu. Jak lze vidět na Obr. 11, systém se skládá z několika front odlišených prioritou, realizovány jsou frontou typu FIFO a pro přístup k výstupnímu portu mají přednostně pakety v zásobníku s nejvyšší prioritou.



Obr. 11: Schéma modelu techniky PQ [7]

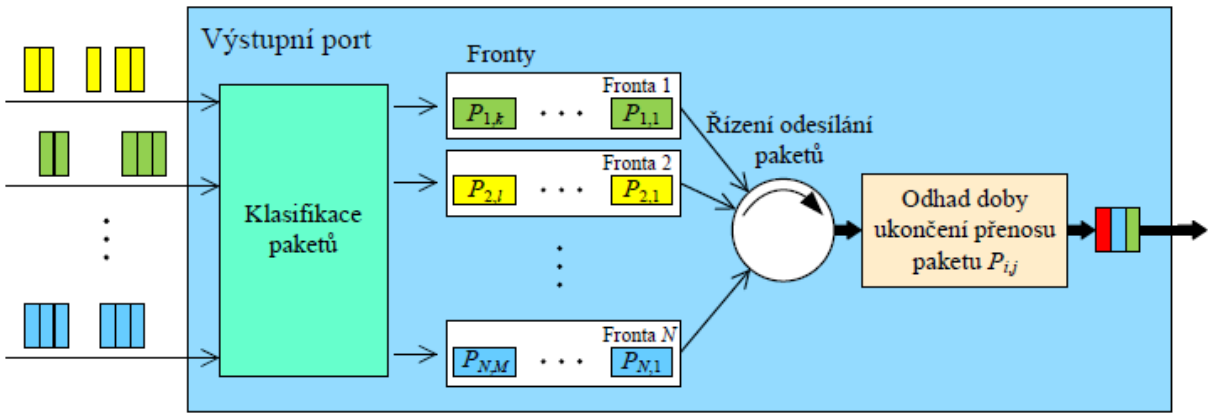
Pakety z fronty s nižší prioritou jsou odeslány až v případě volných front vyšších priorit. Hrozí tedy riziko monopolizace kapacity kanálu jedinou frontou v případě neustálého doplňování paketů a všechny kanály s nižší prioritou nikdy nezískají přístup k výstupnímu kanálu. Při velkém zpoždění těchto paketů mohou být považovány za ztracené a budou znovu poslány, čímž se ještě více zatíží komunikační síť. Efektivně se dají prioritní fronty využít v případě, kdy provoz s vysokou prioritou představuje malou část celkového provozu v síti.

Pro lepší využití komunikačního kanálu je možné použít prioritní frontu s řízenou rychlostí, kdy se šířka pásma pro pakety s nejvyšší prioritou omezí maximální hranicí, kterou tento tok nemůže překročit a i pro fronty nižších priorit zůstane vyhrazena stanovená část šířky pásma. [7]

2.1.3 WFQ

Systém front s váženou spravedlivou obsluhou. Příchozí provoz je dělen do určeného počtu front a každá má přiřazenou váhovou hodnotu, která určuje procento celkové šířky pásma vyhrazené pro frontu. Součet všech váhových koeficientů front odpovídá kompletní šířce pásma. Schéma modelu WFQ je zobrazeno na Obr. 12. Systém obsluhy probíhá cyklicky, a pokud nejsou ve frontě s nejvyšší prioritou připraveny žádné pakety, rozdělí si zbývající fronty kompletní šířku pásma v poměru jejich vah.

Mechanismus WFQ navíc vypočítává teoretickou dobu odesílání, dle které se snaží pakety odesílat. Pro tento výpočet zohledňuje délku paketu, tak aby bylo přidělování šířky pásma opravdu spravedlivé podle přidělených vah. Tím je způsobeno složitější řízení odesílání paketů.



Obr. 12: Schéma modelu techniky WFQ [7]

3 SMĚROVACÍ PROTOKOLY

Směrovací protokoly slouží pro dynamické vytváření směrovacích tabulek popisujících topologii sítě bez nutnosti ručně zapisovat adresy, vyhledávání nejvhodnějších cest podle různých metrik a pravidelné aktualizace všech směrovačů v síti, tak aby byla zaručena konvergence v síti při nastalých změnách nebo výpadcích v co možná nejkratším čase.

V důsledku stále se rozšiřující globální sítě není možné zajistit konvergenci nasazením jednoho směrovacího protokolu, který by spravoval kompletní síť. Naopak vzniká velké množství sítí s různou směrovací politikou a spravované jednou organizací. Takovým sítím se říká autonomní systémy. Z pohledu směrování je základním rozdělením protokolů na EGP (Exterior Gateway Protocol), v současnosti je jediným zástupcem tohoto typu BGP, který slouží na globální úrovni k propojování autonomních systémů, uvnitř kterých jsou nasazeny protokoly IGP (Interior Gateway Protocol). Mezi zástupce patří RIP, RIPv2, RIPng (RIP next generation), OSPF, IS-IS (Intermediate System to Intermediate System), IGRP (Interior Gateway Routing Protocol) a EIGRP (Enhanced Interior Gateway Routing Protocol). [8]

Směrovací tabulku si vytváří každý směrovač v síti a může se lišit dle typu použitého protokolu, nejdůležitějšími položkami jsou: [9]

- původce informace – směrovací protokol, případně manuální
- síťová adresa a maska – definují okruh platných cílových adres pro daný záznam
- identifikace cílové sítě – vypočítaná z masky podsítě cílové adresy
- ohodnocení cesty – závislý na použité metrice
- další skok – síťová adresa sousedního směrovače, který je na cestě k cílové síti
- typ záznamu – statický nebo dynamicky (navíc platnost záznamu)

Protokoly zajišťující směrování uvnitř autonomních systémů se dále dělí dle využívané technologie směrování na dvě hlavní skupiny, Distance-Vector (zástupci jsou RIP, RIP2, IGRP a EIGRP) a Link-State (OSPF, IS-IS).

3.1 RIP

RIP je jedním z nejstarších směrovacích protokolů, a hlavní výhodou je jednoduchost jeho implementace. Má však řadu omezení, kvůli kterým je v současnosti jeho využití omezené. K výběru nejlepší cesty využívá metodu Distance-Vector, která podporuje jediný způsob metriky a tím je počet přeskoků, který odpovídá počtu průchodů aktivním směrovacím prvkem (směrovač). Nedokáže nijak zohlednit parametry spoje, např. propustnost, zpoždění ani neumožňuje manuální ohodnocení jednotlivých cest. Protokol RIP využívá záplavové šíření zpráv obsahujících kompletní směrovací tabulku, a tak navíc nevylučuje možnost vytváření smyček, takové situace mohou nastat například při změně topologie sítě. Pro zamezení tohoto stavu je omezen maximální počet přeskoků na 15, tedy pokud počítadlo přeskoků dosáhne čísla 16, jsou pakety zahazovány a cílová síť je považována za nedostupnou.

Není tak možné použít na rozsáhlejší sítě s větším počtem směrovačů, jelikož by mohlo dojít i k zahazování správně posílaných dat.

Protokol RIP existuje ve dvou verzích:

RIPv1 – 1988, nepodporuje masky sítě, pouze třídní adresování.

RIPv2 – 1998, vylepšená verze, která odstranila nedostatky předchozí verze.

Sestavování směrovacích tabulek probíhá postupným rozesíláním jim známých informací o aktivních rozhraních. Po sestavení kompletní směrovací tabulky ji směrovače rozesílají na všechna rozhraní v pravidelných intervalech (běžně 30 sekund, lze nastavit jiný čas) a porovnávají, zda se přijaté liší od těch vytvořených v paměti. Z toho vyplývá další nevýhoda protokolu RIP, zbytečné vytížení sítě přeposíláním kompletní tabulky i v případě neměnného stavu sítě.

Další nevýhodou je rychlost konvergence, tedy čas potřebný pro aktualizace všech prvků v síti o nastalých změnách. V každém rozhraní směrovače jsou nastaveny tři časovače. Jeden měří pravidelné rozesílání směrovacích tabulek, druhý měří neaktivitu směrovače, pokud směrovač přestane dostávat aktualizace od některého směrovače, běží tento časovač. Pokud dosáhne hodnoty 180 sekund (tedy šesti pravidelných aktualizací) je toto spojení označeno jako nedostupné. Poté se spouští třetí časovač, který obstará vymazání všech záznamů obsahujících cestu nově nedostupnou sítí a ty nejsou dále součástí směrovacích tabulek. K tomu dojde, až napočítá časovač hodnotu 240 sekund. [10]

3.2 OSPF

Protokol je typu Link-State, využívá k vypočtení nejvhodnější cesty pokročilý systém metrik, do kterých lze započítat parametry dané linky (šířka pásma, zpoždění, spolehlivost, cena) a umožňuje více možností nastavení metriky pro tento výpočet než protokol RIP. Rozsah čísla pro ohodnocení cesty může nabývat hodnot od 0 až do 65535. [11] K optimalizaci zvolené trasy slouží algoritmus SPF (Shortest-Path-First) a Dijsktrův algoritmus, který vybírá nejlepší cestu výpočtem minimalizace celkového součtu metrik. Nevýhodou je složitější implementace protokolu, první verze protokolu vznikla v roce 1991 a v roce 1999 přibyla verze OSPFv3 s podporou pro sítě s adresací IPv6. [12]

Protokol OSPF je vhodný i pro směrování v rámci rozsáhlejších sítí, ale s rostoucí velikostí sítě se zvětšují nároky na výpočty cest a také časy konvergence i vytížení sítě rozesíláním aktualizací. Jelikož pro výpočet nejkratší cesty musí každý směrovač znát kompletní topologii sítě. Vhodným řešením je rozdělení sítě do menších oblastí a routery uvnitř oblastí poté pracují pouze s informacemi o zařízeních ve stejné oblasti. O směrování do sousedních oblastí se starají hraniční routery, které propojují dvě, anebo více oblastí a musí tak mít znalost topologie všech připojených oblastí.

Tyto informace však nepřeposílají mezi jednotlivými oblastmi, a tak dochází ke snížení provozu v síti i výpočetních nároků směrovačů. Oproti protokolu RIP také nabízí rychlejší konvergenci

K šíření informací mezi směrovači je využíváno záplavové šíření zpráv multicastovým přenosem, protokol OSPF používá pro sestavení směrovacích tabulek několik definovaných typů zpráv: [4]

- Hello – jsou vysílány všemi aktivními porty do sousedních směrovačů, slouží k objevování sousedů a k ověřování platnosti spojení mezi sousedními směrovači. Odesílány jsou v pravidelných intervalech, typicky po 10 sekundách. Pokud se router nehlásí (nedorazí od něj žádná hello zpráva), je po uplynutí časového intervalu (tzv. dead-interval) označen jako nedostupný, typická hodnota je 40 sekund.

- DBD (The Database Description) - zkrácený výpis databáze vysílajícího směrovače, slouží pro rychlejší sestavení směrovací tabulky nově připojeného routeru do sítě, který si z databáze vybere položky, které mu chybí a následně zažádá o jejich kompletní informace.

- LSR (Link-State Request) - žádost o informace z databáze DBD.

- LSU (Link-State Update) - jsou odpovědí na přijaté žádosti LSR. Obsahují zprávy LSA (Link-State Advertisement), které popisují stav rozhraní nebo seznam směrovačů připojených k dané síti. Tyto zprávy si ukládají do vlastní databáze a přeposílají je dále na přímo připojené směrovače. Po sestavení kompletní topologie sítě provádí každý směrovač samostatně výpočet nejlepších cest pomocí Dijkstrova alogritmu. Poté již routery neposílají tento typ zpráv až do doby, než dojde ke změně topologie. V takovém případě rozesílá směrovač, na kterém došlo ke změně novou zprávu LSA a probíhá znovu výpočet nejkratších cest.

- LSAck (Link-State Acknowledgement) – potvrzení o přijetí LSU.

4 PŘENOSOVÉ TECHNOLOGIE LOKÁLNÍCH SÍTÍ

Pod pojmem lokální síť se v oblasti komunikačních technologiích rozumí propojení skupiny uživatelů, umožňující jejich vzájemnou komunikaci a přenos dat, v prostorově omezeném rozsahu, maximálně v řádu kilometrů. Typicky se jedná o síť provozovanou v rámci jediné budovy a k jejich souhrnnému pojmenování se užívá zkratka LAN (Local Area Network). Přenosové rychlosti mohou dosahovat až 100 Gbit/s, jednotlivé technologie jsou definovány komplexní skupinou standardů IEEE 802, z nichž v současnosti nejpoužívanějšími jsou standard IEEE 802.3, popisující kabelové spoje a standard IEEE 802.11 popisující bezdrátové spojení s přenosem dat radiovým signálem.

4.1 IEEE 802.3 – Ethernet

Síť Ethernet popsaná standardem IEEE 802.3 je v současnosti nejvyužívanějším typem pro realizaci lokálních počítačových sítí. Ethernet byl navržen v roce 1973 společností Xerox a původní přenosová rychlost byla 2,94 Mbit/s po koaxiálním kabelu [11]. Během let prošel rozsáhlým vývojem a bylo navrženo mnoho různých verzí s postupným vylepšováním technických parametrů, z nichž nejpodstatnějším je přenosová rychlost. Přehled jednotlivých fází vývoje zobrazuje Tabulka 1. Standardizovaná síť Ethernet zajišťuje kompatibilitu s širokou škálou výrobců a použitelnost s mnoha typy síťových zařízení. Kromě běžných síťových prvků (rozbočovač, přepínač, směrovač, most, klientské stanice) lze do sítě Ethernet připojit například IP telefony, telefonní servery nebo tiskárny.

Z technologického hlediska se Ethernet dělí na dva typy podle způsobu přenosu informace, jimiž je využití elektrických jevů v metalických vodičích nebo optických jevů v optických vláknech.

V metalických vodičích jsou přenášené bity zakódovány pomocí napěťových úrovní, které jsou na přijímací straně převedeny zpět na bity s hodnotou jedna nebo nula. Zvolený typ kódování a z něj určené hodnoty napěťových úrovní se liší dle použitého standardu. Fyzická vrstva dále zajišťuje synchronizaci sériového toku bitů, detekci obsazení kanálu, detekci kolize a detekci nosné. V současnosti se k přenosu nejčastěji používá nestíněná kroucená dvoulinka s označením UTP (Unshielded Twisted Pair), která umožňuje komunikaci v plně duplexním módu, tedy současné komunikaci obou stran, jelikož je pro vysílací i přijímací stranu vyhrazen samostatný pár vodičů a nehrozí tak vznik kolizí. Původní standard byl realizován koaxiálním kabelem, který neumožňuje oddělení vysílací a přijímací strany, komunikace mohla probíhat pouze v poloduplexní komunikaci a bylo nutné použití přístupové metody CSMA/CD (Carrier Sense Multiple Access with Collision Detection) pro řízení přístupu k médiu. Nutnost použití se s postupným vývojem omezila a v současnosti již není užívána, prakticky všechny sítě jsou realizovány jako plně duplexní a přenosová rychlost odpovídá maximální teoretické rychlosti. K zapojení kabelů se používá osmi-pinový konektor s označením RJ-45, podle druhu propojovaných zařízení se užívá přímý nebo překřížený kabel.

V optických vodičích se k přenosu informace využívá světla produkovaného zdrojem záření na koncích vedení, využívá se záření v oblasti vlnových délek 850 nm, 1300 nm a 1550 nm. Oproti metalickým vodičům mají optická vlákna nižší hmotnost a vysokou odolnost vůči vnějšímu rušení, díky nízkému útlumu signálu je možné realizovat velmi dlouhá vedení, řádově až desítky km bez regenerátoru signálu. Vlákna mohou být jednovidová, ta mají obrovskou přenosovou kapacitu a dosah signálu, ale náklady na výrobu jsou výrazně vyšší než pro vlákna mnohovidová, ta jsou nejčastěji používána v běžných aplikacích pro komunikaci na kratší vzdálenosti.

Standard IEEE 802.3 kromě fyzické vrstvy popisuje také vrstvu spojovou, která je druhou vrstvou referenčního modelu ISO/OSI. Datová jednotka na této vrstvě se nazývá rámec a má pevně definovanou strukturu. Minimální velikost rámce je 64B a maximální velikost 1518B a musí obsahovat povinná pole, jimiž jsou cílová a zdrojová fyzická adresa, každá má 6B, dále pole, o velikosti 2B, definující velikost datového pole nebo určující typ požadovaného protokolu vyšších vrstev. Část obsahující přenášená data může mít velikost v rozsahu od 46B-1500B, při menší velikost dat musí být použita výplň k doplnění do 46B, aby byla splněna podmínka minimální velikosti rámce. Konečným polem je kontrolní součet o velikosti 4B, který detekuje chyby vzniklé přenosem. Na fyzické vrstvě se před každý rámec přidá pole o velikost 8B, z nichž prvních 7 oktetů obsahuje střídající se jedničky a nuly, které slouží k synchronizaci a další oktet upozorňuje na začátek nadcházejícího rámce. [13]

Tabulka 1: Přehled standardů a typů sítí Ethernet [11]

Rychlost sítě	**Standard IEEE**	**Označení typů**	**Rok vývoje**
10 Mbit/s	802.3	10Base-2, 10Base-5, 10Base-T, 10Base-FL, 10Base-FB, 10Broad-36	1979-1993
100 Mbit/s	802.3u	100Base-TX, 100Base-T4, 100Base-FX	1995
	802.3xy	100Base-T2	1997
1 Gbit/s	802.3z	1000Base-LX, 1000Base-SX, 1000Base-CX	1998
	802.3ab	1000Base-T	2000
10 Gbit/s	802.3ae	10GBase-SR, 10GBase-SW, 10GBase-LX4, 10GBase-LR, 10GBase-LW, 10GBase-ER, 10GBase-EW	2002
	802.3ak	10GBase-CX4	2003
	802.3an	10GBase-T	2006
	802.3ap	10GBase-KR, 10GBase-KX4	2007
40 Gbit/s 100 Gbit/s	802.3ba	40GBase-KR4, 40GBase-CR4, 40GBase-SR4, 40GBase-LR4, 100GBase-CR10, 100GBase-SR10, 100GBase-LR4, 100GBase-ER4	2010

4.1.1 Vývoj standardu IEEE 802.3

Ethernet

Někdy také označovaný jako standartní Ethernet, podporuje přenosovou rychlost 10 Mbit/s, pracuje jako sdílené médium a je tedy nutná implementace přístupové metody CSMA/CD pro minimalizace kolizí. Jako přenosové médium lze použít koaxiální kabel, nestíněnou kroucenou dvoulinku s jedním párem vodičů či optický kabel, poslední jmenovaný umožňuje maximální délku linky až 2000 m [10].

Fast Ethernet

Zpětně kompatibilní s předchozí generací, maximální přenosová rychlost se zvýšila na 100 Mbit/s. Kvůli požadavku zpětné kompatibility zůstává stejný formát rámce, a také implementace metody CSMA/CD, která je použita v poloduplexním módu. Tato generace již umožňuje plně duplexní mód, pro který odpadá nutnost použití přístupové metody. Nově je implementována funkce pro automatické dohodnutí parametrů přenosu mezi komunikujícími zařízeními, ty si sami dohodnou přenosovou rychlost a duplexní mód. Praktická realizace je možná pomocí stíněné kroucené dvoulinky, optického vlákna, pro obě technologie s použitím jednoho páru vodičů/vláken. Dva páry vodičů lze použít pouze s nestíněnou kroucenou dvoulinkou.

Gigabit Ethernet

Přebírá funkce z předchozích generací a opět je zajištěna zpětná kompatibilita, přenosový rychlost vzrostla na 1 Gbit/s. Při použití optických vláken s dlouhou vlnovou délkou lze realizovat linku o délce až 5000 m.

Ten-Gigabit Ethernet

Opět je zajištěna podpora zpětné kompatibility, formát rámce je zachován a přebrány jsou i funkce předchozích generací, ovšem již není implementována přístupová metoda CSMA/CD a je možné pouze použití plně duplexního módu. Nově je umožněno propojení lokálních sítí s MAN (Metropolitan Area Network) a WAN (Wide Area Network) sítěmi. Umožněna je také kompatibilita s technologiemi ATM (Asynchronous Transfer Mode) a Frame Relay. Lze realizovat linky o velmi dlouhé délce, až 40 km, pomocí dvojice jednovidových optických vláken. [10]

40G/100G Ethernet

Zajištěna zpětná kompatibilita, stejně jako předchozí generace podporuje pouze plně duplexní mód, nově je podporována volitelná funkce FEC (Forward Error Correction) pro dopřednou opravu chyb. Technologický standard s přenosovou rychlostí 40 Gbit/s slouží k propojení datových center a verze s rychlostí 100 Gbit/s je používána k propojení přepojovacích prvků. [11]

4.2 IEEE 802.11 - WLAN

Bezdrátové sítě jsou v principu totožné se sítěmi kabelovými, jenom se změnou přenosového média, kterým je volně šířený radiový signál. Ovšem tento rozdíl je velmi podstatný a ovlivňuje způsob, jakým jsou sítě budovány, a jaké podmínky jsou kladeny na standardy definující potřebné technologie. Nejrozšířenějším standardem popisujícím technologie bezdrátových lokálních sítí se stal IEEE 802.11, jeho původní verze byla vyvinuta v roce 1997 a během let prochází neustálým vývojem a jsou vyvíjeny nové modifikace tohoto standardu, nejpoužívanější standardy zobrazuje Tabulka 2.

Bezdrátové sítě mají řadu výhod, podstatnou z nich je dynamická topologie, která umožňuje přesun, přidávání a odebírání komunikujících stanic bez nutnosti zásahu do topologie a nastavení sítě. Je možné provozování bezdrátové sítě v oblastech, kam není vhodné či je úplně nemožná instalace kabelového vedení.

Pro bezdrátové sítě nelze pevně definovat a ani vysledovat hranice, kde daná síť a její signál končí, topologie bezdrátových sítí jsou dynamické a přenášený signál je podstatně více náchylný k okolnímu rušení. To vše vyplývá ze základního faktu, že každá stanice nebo síťový prvek vysílá signál všesměrově a to do vzdáleností definovanými okolním prostředím a výkonem použitého vysílače/přijímače. V praxi to znamená, že jednotlivé sítě nacházející se ve vzájemném dosahu se mohou překrývat a vzniká tak možnost kolizí, přeslechů a celkově degradace signálu, případně parametrů přenosu. Bezdrátové sítě tedy lze obecně brát jako méně spolehlivé a tyto faktory je nutné brát v potaz při návrhu sítě. Vzhledem ke sdílení přenosového kanálu jsou kladeny vysoké nároky na zabezpečení přenosu a zavedení přístupových metod. Bezdrátové sítě jsou vhodné pouze pro realizaci koncových sítí, jelikož dosah signálu je obecně v řádech desítek metrů až stovky metrů a s rostoucí vzdáleností navíc dochází ke zhoršování parametrů přenosu (vyšší chybovost, snížená přenosových rychlost).

4.2.1 Vývoj standardu IEEE 802.11

Původní standard 802.11 je považován za zastaralý a praktický se nepoužívá, vylepšením byl standard s označením 802.11b, který zvýšil maximální přenosovou rychlost na 11 Mbit/s, podporovány byly také rychlosti 1 Mbit/s, 2Mbit/s a 5,5 Mbit/s. Pro každou rychlost je užíváno jiné kódování a s rostoucí rychlostí se také snižuje odolnost proti rušení a tím vznik chyb, výběr rychlost je prováděn automaticky podle změřené chybovosti. [14]

Standard 802.11a se liší využitím pásma 5Ghz, které je v Evropě rozděleno na 19 kanálů o šířce 16,6 MHz a s dostupem 20 MHz. Oproti pásmu 2,4 GHz se vyznačuje menší intenzitou rušení, vzhledem k menší obsazenosti v oblasti tohoto kmitočtu, ale nevýhodou je menší dosah signálu. Přenosová rychlost dosahuje až 54 Mbits/s.

Standard 802.11g přebírá způsob modulace a přenosové rychlosti od standardu 802.11b, se kterým je zpětně kompatibilní a navíc umožňuje rychlosti až do 54 Mbit/s, pro které je použita modulace OFDM (Orthogonal Frequency Division Multiplexing).

Výraznou inovací přispěl standard 802.11n, který zavádí koncept MIMO (Multiple Input Multiple Output), ten umožňuje použití několika antén ve vysílací i přijímací části, každá z nich

může vysílat různém, prostorově odděleném kanále a každým kanálem (maximálně čtyři) lze přenášet samostatný datový tok. Přenosová rychlost jednoho kanálu je až 150 Mbit/s, s použitím technologie MIMO tedy teoretická hodnota dosahuje až 600 Mbit/s.

Standard 802.11ac vychází z předchozí verze, kterou vylepšuje možností vysílání až osmi kanály s technologií MIMO, každý kanál má maximální přenosovou rychlost 866,7 Mbit/s. [14]

Tabulka 2: Nejpoužívanější standardy IEEE 802.11 [14]

Označení	Vznik	Max. přenosová rychlost [Mbit/s]	Použité frekvenční pásmo [GHz]
802.11	1997	2	2,4
802.11b	1999	11	2,4
802.11a	1999	54	5
802.11g	2003	54	2,4
802.11n	2009	150	2,4 nebo 5
802.11ac	2014	866,7	5

Základním prvkem pro realizaci bezdrátových sítí je přístupový bod, který je obdobou přepínače v sítích Ethernet. Přístupový bod zajišťuje přenos komunikace mezi všemi stanicemi v dané síti, a také dokáže spojovat bezdrátovou síť se sítí Ethernet a jejími stanicemi. Přístupový bod také zajišťuje možnost připojení stanic v dosahu jeho signálu, k tomu jsou určeny rámce správy, které jsou rozesílány přístupovým bodem v pravidelných intervalech (typicky každých 100 ms) a nazývají se „beacon“ rámce. V nich sděluje informace o propagované síti, parametry a požadavky, podle kterých se mohou stanice do sítě připojit. Důležité části beacon rámce jsou časové značky, které slouží k synchronizaci stanic, podporované přenosové rychlosti, způsob zabezpečení a název sítě SSID (Service Set Identifier). Stanice po vstupu do oblasti sítě naslouchá těmto rámcům a poté na něj odpovídá se žádosti o autentizaci, pokud se stanice dohodne s přístupovým bodem na parametrech (společná přenosová rychlost, typ zabezpečení, heslo) dochází k připojení do sítě. Pokud stanice obdrží vícero beacon rámců vybírá si mezi podle parametrů propagovaných přístupovým bodem nebo dle síly signálu.

K řízení přístupu ke sdílenému médiu je implementována metoda CSMA/CA (Carrier Sense Multiple Acces with Collision Avoidance). Stanice před zahájením vysílání naslouchá, po dobu náhodně zvolenou z daného intervalu, pokud je během této doby stanice neaktivní, tak po uplynutí další čekací doby DIFS (Distributed Interframe Space) odešle rámec s žádostí k odesílání RTS (Request To Send). Po úspěšném uplynutí dalšího intervalu s názvem SIFS (Short Interframe Space) odešle cílová stanice odpověď s povolením k přenosu CTS (Clear To Send). Po uplynutí dalšího intervalu SIFS může začít stanice vysílat data.

Jsou definovány tři typy rámců, řídící, datové a rámce správy. Poslední jmenované slouží k navázání vzájemné komunikace, řídící rámce obstarávají přístup ke kanálu a datové rámce obsahují data a kontrolní informace. Všechny mají pevně definovaný formát, každý rámec na

linkové vrstvě se skládá z devíti částí. První pole FC (Frame Control) o velikosti 2B definuje typ rámce a další řídící informace, druhé pole D o velikost 2B definuje délku trvání přenosu, s výjimkou řídících rámců, pro které určuje jejich identifikační číslo. Čtyři pole s MAC (Media Access Control) adresami stanic, které se podílejí na přenosu, každé pole má velikost 6B. Další pole je pořadové číslo rámce SC (Sequence Control) o velikosti 2B. Další částí jsou samotná data s velikosti od 0B do 2312B (velikost a obsah tohoto pole se liší dle typu rámce), posledním polem je pole FCS o velikosti 4B a obsahuje kontrolní součet pro detekci chyb. [13][15]

4.2.2 Zajištění kvality služeb

Se stále rostoucí oblibou bezdrátových sítí, se také zvyšovaly požadavky na zajištění kvality služeb. Zpočátku nebylo při vývoji standardů věnováno příliš pozornosti, což bylo napraveno vývojem standardu IEEE 802.11e, který rozšiřuje přístupové mechanismy původních WLAN sítí a nově umožňuje podporu mechanismů pro zajištění kvality služeb, Qos. Návrh byl propracován tak, aby byla zachována kompatibilita se zařízeními, které tento standard nepodporují. Nejdůležitějšími metodami, které standard IEEE 802.11e zavádí, jsou nové koordinační funkce, těmi jsou distribuovaná koordinační funkce EDCF (Enhanced Distributed Coordination Function) a hybridní koordinační funkce HCF (Hybrid Coordination Function), kterých využívají nově definované metody přístupu ke kanálu HCCA (HCF Controlled Channel Access) a EDCA (Enhanced Distributed Channel Access). [7]

Metoda přístupu ke kanálu EDCA

Tento mechanizmus pro zajištění kvality služeb nabízí rozdělení datového provozu do čtyř kategorií, a pro každou kategorii navíc může být definováno několik priorit, kterých může být celkově až osm. Jednotlivé kategorie jsou označeny jako Background (data pozadí), Best Effort (základní přenos), Video (obrazový signál) a Voice (přenos hlasu). Při současném vysílání dat v různých kategorií, dochází k soutěžení o přenosový kanál, princip vychází z původních přístupových metod, které aplikují stanice při žádosti o vysílání a využívají čekací rámce SIFS. Navíc se zavádí nový čekací rámec AIFS (Arbitration Interframe Space), jehož délka se liší podle kategorie provozu a další možností diferenciace je nastavení minimální a maximální hodnoty parametru CW (Contention Window), při nižší hodnotě je pravděpodobnost přístupu ke kanálu vyšší. Pokud komunikuje stanice s podporou mechanismu EDCA se stanicí, která jej nepodporuje, je provoz řazen automaticky do kategorie Best Effort, stejně tak je tomu v případě, že se jedná o datový tok nepřiřazený žádné z definovaných kategorií. [15]

5 AKTIVNÍ SÍŤOVÉ PRVKY

Způsob funkce a principy, které jsou aplikovány při zpracování a přenosu zpráv v rámci počítačových sítí vychází z rozdělení jednotlivých úkonů do skupin definovaných v referenčním modelu ISO/OSI, případně ve zjednodušeném modelu TCP/IP. Složení obou modelu zobrazuje Tabulka 3. Pro úspěšné provedení a zpracování přenosu informace, dochází k postupnému zpracování datové jednotky danou vrstvou a předání její sousední vrstvě. Pro samotný přenos dat jsou nejdůležitější tři nejnižší vrstvy, fyzická, spojová a síťová. Zbylé čtyři vrstvy, respektive aplikační vrstva modelu TCP/IP mají za úkol především práci s datovým obsahem zpráv a jsou obvykle záležitostí uživatelských aplikací.

Na třech nejnižších vrstvách operují aktivní síťové prvky, které zprostředkovávají přenos informace skrze komunikační síť. Podle typu zařízení mohou vykonávat i mnoho dalších funkcí, jako řízení přístupu ke sdílenému médiu, zajištění kvality služeb nebo směrování datového toků sítí s nalezením nejlepší cesty. Množina operací, které jsou schopny dané prvky provádět, vychází právě z principu rozdělení komunikačního modelu na jednotlivé vrstvy.

Síťová vrstva a na ní operující síťové prvky tedy mohou poskytnout nejrozsáhlejší možnosti při zpracování přijatých nebo odesílaných zpráv, jelikož prvky na této vrstvě musí také zvládat operace obou nižších vrstev. Nevýhodou jsou vyšší výpočetní nároky, které vyplývají z vyšší složitosti prováděných operací, a také riziko nefunkčnosti, či znehodnocení parametrů komunikační sítě vzniklé nevhodným nastavením. Vzhledem k vyšším možnostem režie datového toku, také vzniká možnost nárůstu časového zpoždění přenášených dat. Stejný princip platí i pro aktivní síťové prvky druhé vrstvy, tedy zpracování datových jednotek probíhá na fyzické a spojové vrstvě. Tyto prvky tedy nabízí menší možnosti nastavení a řízení provozu, ale výhodou jsou nižší výpočetní nároky. Síťové prvky nejnižší vrstvy pracují pouze s vrstvou fyzickou a jejich možnosti jsou tedy velmi omezené, výhodou je však jednoduchost nastavení, minimální výpočetní nároky a minimální zpoždění přenášených dat.

Tabulka 3: Referenční model ISO/OSI a TCP/IP

<table>
<tr><td>7. Aplikační vrstva</td><td rowspan="5">4. Aplikační vrstva</td></tr>
<tr><td>6. Prezentační vrstva</td></tr>
<tr><td>5. Relační vrstva</td></tr>
<tr><td>4. Transportní vrstva</td></tr>
<tr><td>3. Síťová vrstva</td><td>3. Síťová vrstva</td></tr>
<tr><td>2. Spojová vrstva</td><td>2. Spojová vrstva</td></tr>
<tr><td>1. Fyzická vrstva</td><td>1. Fyzická vrstva</td></tr>
</table>

5.1 Fyzická vrstva

Nejnižší vrstva referenčního modelu ISO/OSI, na které se operuje se signálem reprezentovaným ve fyzické podobě, v kabelových sítích se jedná o signál elektrický. Tato vrstva zajišťuje přijímání a vysílání signálu komunikační sítí a kromě fyzických parametrů signálu definuje také kódování, přenosovou rychlost a fyzické propojení jednotlivých prvků sítě, tedy kabeláž, zapojení konektorů.

Jelikož se jedná pouze o fyzickou vrstvu, nijak se nepracuje s adresami, ať už fyzickými či logickými ani s obsahem přenášených dat. Jediné operace, které mohou aktivní prvky provádět, se týkají modifikace fyzického signálu, napěťové úrovně, tvar nebo časování.

Základním prvkem fyzické vrstvy jsou opakovače, které mají jeden vstupní a jeden výstupní port. Dokáží zpracovávat signál pouze pro jeden směr komunikace a pro použití v obousměrné komunikaci musí být použit opakovač pro každý směr. Jejich hlavní funkcí je obnova, případně zesílení přenášeného signálu, tak aby bylo možné přenášet data na delší vzdálenosti, proto je oblast využití především v geograficky rozsáhlých sítích.

Rozbočovač vychází z principu funkce opakovače, ovšem obsahuje několik portů a všechny jsou obousměrné. Na všech portech musí být data přenášena stejnou rychlostí. Pracuje pouze se signálem a jednotlivými bity, které z příchozího portu posílá na všechny ostatní porty, vzniká tak hvězdicová topologie, kdy rozbočovač působí jako centrální prvek [16]. Rozbočovač se již v dnešní době příliš nepoužívá a obzvlášť v sítích s větším počtem stanic propojených rozbočovači dochází k vysokému zatížení sítě vlivem všesměrového rozesílání signálu. Navíc vzniká riziko kolizí při současném vysílání dvou a více stanic, při nich vznikají neplatné rámce, které jsou dále rozesílány na všechny stanice v síti a dochází ke zbytečnému zahlcování sítě. Vzhledem k absenci jakéhokoliv filtrování datového provozu, také existuje riziko vzniku smyček obzvláště při rozsáhlejší a složitější topologii sítě.

5.2 Spojová vrstva

Druhá vrstva referenčního modelu ISO/OSI, která již překládá fyzický signál na datové jednotky, které se nazývají rámce. Ty neobsahují pouhá data, ale také směrovací informace, zdrojovou a cílovou fyzickou adresu, kontrolní součet a další položky. Podrobný popis rámce linkové vrstvy sítě Ethernet je uveden v kapitole 4.1. Z adres obsažených v každém rámci lze identifikovat jednotlivá spojení, a provádět filtrování přijatých dat.

Hlavním aktivním síťovým prvkem operujícím na spojové vrstvě je přepínač. Jehož základní funkcí je přepínání příchozích datových jednotek na odpovídající výstupní porty. Volba výstupního portu je určena tabulkou fyzických adres, kterou si každý přepínač vytváří dynamicky, čtením pole zdrojové adresy v příchozích rámcích. Přepínač navíc rozděluje kolizní doménu, díky čtení a kontrole obsahu jednotlivých polí přijatých rámců, nejsou přepínači dál přeposílány chybné rámce.

Databáze adres stanic připojených k portům přepínače obsahuje kromě dvojice adresa, port, také časovač pořízení záznamu. Ten slouží k odstranění záznamu z tabulky po určitém

čase, pokud stanice s danou adresou byla odpojena nebo vypnuta a tedy již dále nekomunikuje. Pokud není v tabulce uložena cílová adresa přijatého rámce, tak přepínač rozesílá data na všechny porty, kromě toho, ze kterého je přijal. [16]

5.3 Síťová vrstva

Třetí vrstva referenčního modelu ISO/OSI zajišťuje cestu datových jednotek od zdroje až k cíli, to je umožněno směrováním datových jednotek sítí, které jsou posílány cestou vybranou jako nejvhodnější. Určení cestu se může lišit a je závislé především na použitém směrovacím protokolu a topologii sítě. Přehled směrovacích protokolu je uveden v kapitole 3. Datová jednotka, se kterou se pracuje na síťové vrstvě, se nazývá paket, nebo také IP datagram.

Aktivním prvkem na síťové vrstvě je směrovač, který určuje optimální cestu mezí zdrojem a cílem přijaté zprávy. To je rozhodováno na základě směrovací tabulky, kterou si uchovává každý směrovač. Vytváření směrovacích tabulek může probíhat dvojím způsobem. Statické vytváření tabulek provádí uživatel/správce sítě ručním zadáním jednotlivých záznamů, tento způsob je vhodný pouze pro jednoduché sítě se stálou topologií, protože každá změna v síti musí být pro správnou funkčnost komunikace reflektována do směrovacích tabulek každého směrovače, což je nutno provést manuální úpravou. Ve většině případů je vhodnější dynamický způsob vytváření směrovacích tabulek, v takovém případě si informace o síti vyměňují směrovače mezi sebou a postupně tak vytvářejí mapování kompletní sítě. Dynamické směrování tak dokáže reagovat na změny v topologii, pokud například dojde k poruše směrovače v síti, ostatní směrovače dokáží poměrně rychle reagovat a nalézt náhradní cestu. Rychlost, s jakou jsou všechny směrovače v síti schopny aktualizovat obsah směrovacích tabulek po změně topologie, se nazývá konvergence. Rychlost konvergence je závislá na typu a nastavení použitého směrovacího protokolu, může se pohybovat řádově v jednotkách sekund. Nevýhodou je vzájemné rozesílání informací mezi směrovači, které tak snižuje kapacitu přenosových linek.

Každý platný záznam směrovací tabulky musí obsahovat IP adresu cílové sítě a masku této sítě, IP adresu dalšího skoku (směrovače na cestě k cílové síti) a číslo rozhraní, kterému jsou tyto údaje přiřazeny. Pro dynamické směrování je v tabulce navíc pole určující metriku, neboli cenu daného spoje a pole se stářím zápisu daného záznamu, které hlídá jeho platnost. Po vypršení nastavených časových úseků je záznam považován za neplatný a poté může dojít k jeho vymazání.

Směrovače také umožňují překlad logických adres na adresy fyzické, dokáží spojovat rozhraní s různými přenosovými rychlostmi a podporují mechanismy pro řízení kvality služeb.

6 PRAKTICKÁ REALIZACE ZADÁNÍ

Tato kapitola obsahuje stručné shrnutí jednotlivých laboratorních úloh a cílů, kterých má být dosaženo při vypracování dané úlohy. Každá úloha také poskytuje zobrazení topologie sítě, pro kterou jsou simulace prováděny.

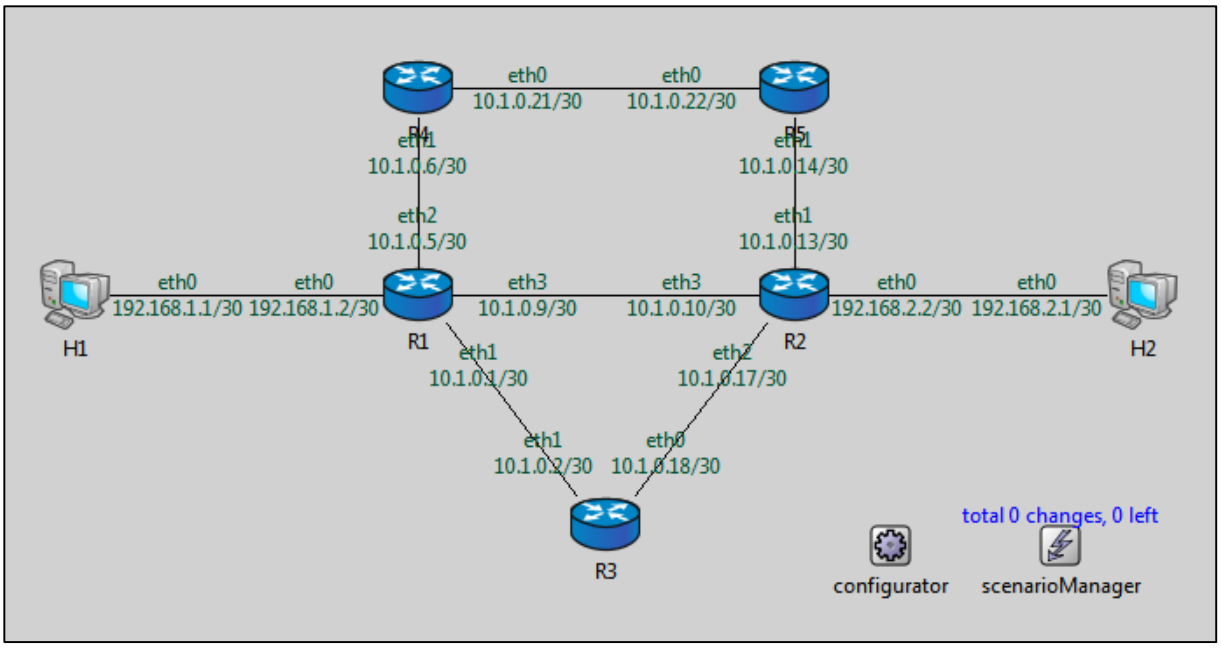
6.1 Laboratorní úloha č.1 – Směrovací protokoly

Cílem této laboratorní úlohy je srovnání vlastností směrovacích protokolů RIP a OSPF. Simulace bude prováděna v jednoduché síti s pěti směrovači a dvěma koncovými zařízeními propojenými technologií ethernet. Topologie zadání je na Obr. 13.

První část úlohy zahrnuje vytvoření dvou sítí se zavedeným směrovacím protokolem RIP respektive OSPF. Úkolem je sledování procesu vytváření směrovacích tabulek v simulačním nástroji OMNeT++/Tkenv a srovnání záznamů ve výsledných tabulkách obou protokolů. Po dokončení simulace lze tento proces dále analyzovat ve výsledcích.

Dalším úkolem je rozdělení sítě s protokolem OSPF na dvě menší oblasti a porovnání s nerozdělenou sítí.

Pro další část úlohy se nastaví generování síťového provozu z koncové stanice H1 ke stanici H2 a vytvoří se scénář s výpadkem linky a následným obnovením, který umožní porovnání reakce na změnu topologie obou protokolů a volbu náhradní cesty k cílové síti. Kompletní návod k laboratorní úloze je k nalezení v příloze A.



Obr. 13: Topologie sítě pro směrovací protokoly

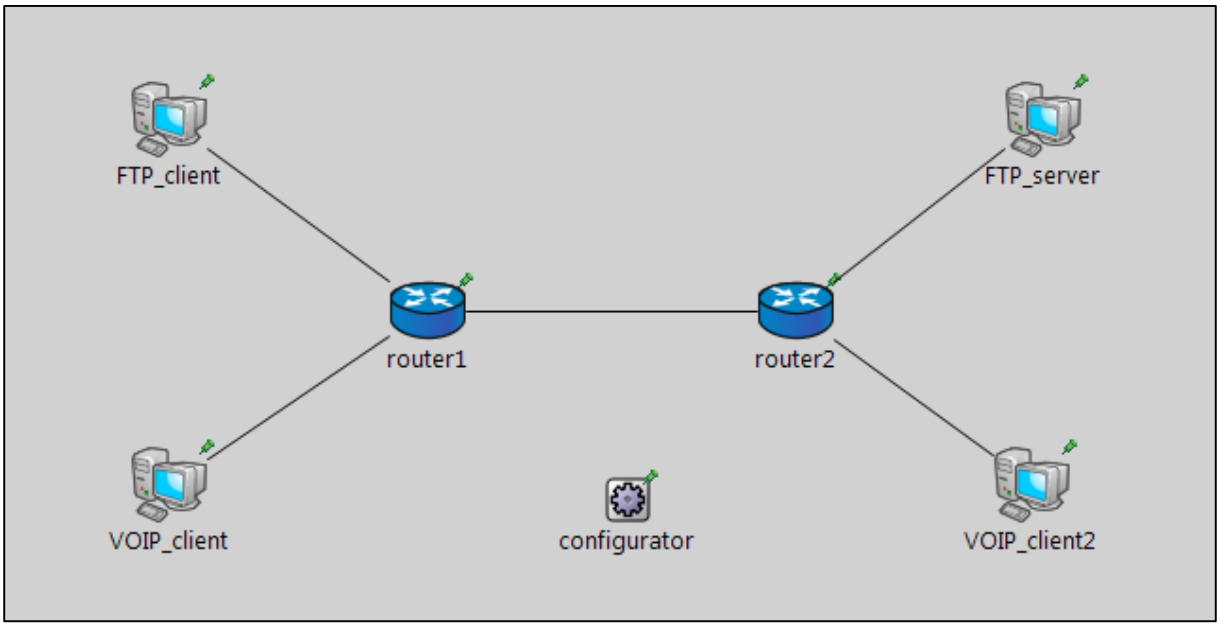
6.2 Laboratorní úloha č.2 – Zajištění kvality služeb

Cílem této laboratorní úlohy je srovnání různých mechanizmů řízeného odesílání paketů a jejich vlivu na parametry definující kvalitu síťového přenosu (zpoždění, proměnlivost zpoždění, ztrátovost paketů). K simulaci poslouží jednoduchá síť s linkou poskytující nedostatečnou šířku pásma pro definovaný provoz a koncové stanice současně generující FTP (File Transfer Protocol) datový přenos a VOIP hovor. Zapojení topologie sítě lze vidět na Obr. 14.

První částí úlohy je vytvoření sítě a všech souborů potřebných pro realizování mechanizmů pro řízení odesílání paketů. V úloze jsou tři různé scénáře, bez zavedených technik QoS (fronta FIFO), mechanizmus prioritních front s řízenou rychlostí PQ a mechanizmus spravedlivých váhových front WFQ.

Po vytvoření sítí následuje simulování všech vytvořených scénářů a porovnání získaných výsledků. Ze získaných hodnot jsou důležité hlavně parametry přenosu VOIP hovoru, průměrné zpoždění a kolísání zpoždění (jitter). A také počet a druh zahozených paketů a obousměrné zpoždění (RTT – Round-Trip Time) přenosu FTP paketů.

Následně se vyzkouší vliv nastavení parametrů jednotlivých mechanizmů pro řízení provozu a jejich změn na výslednou kvalitu služeb. Kompletní návod k laboratorní úloze je k nalezení v příloze B.

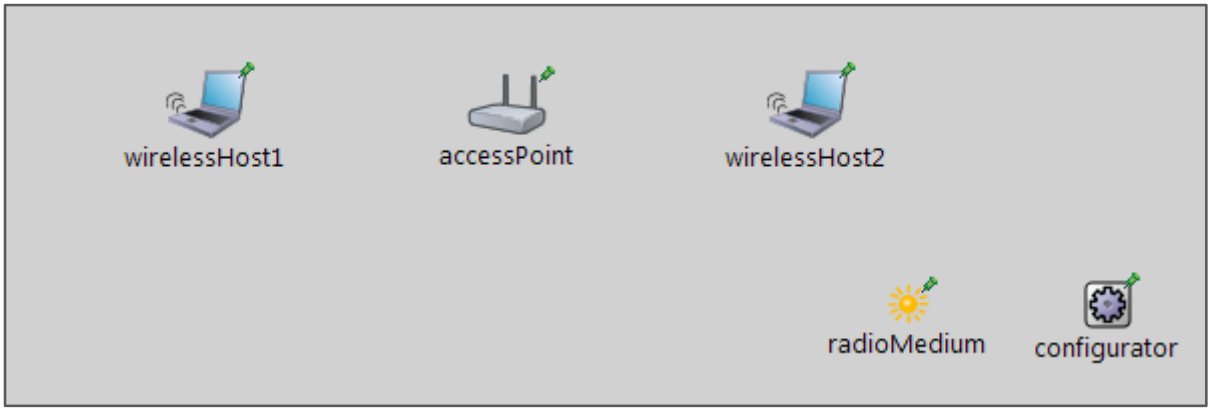


Obr. 14: Zapojení topologie sítě

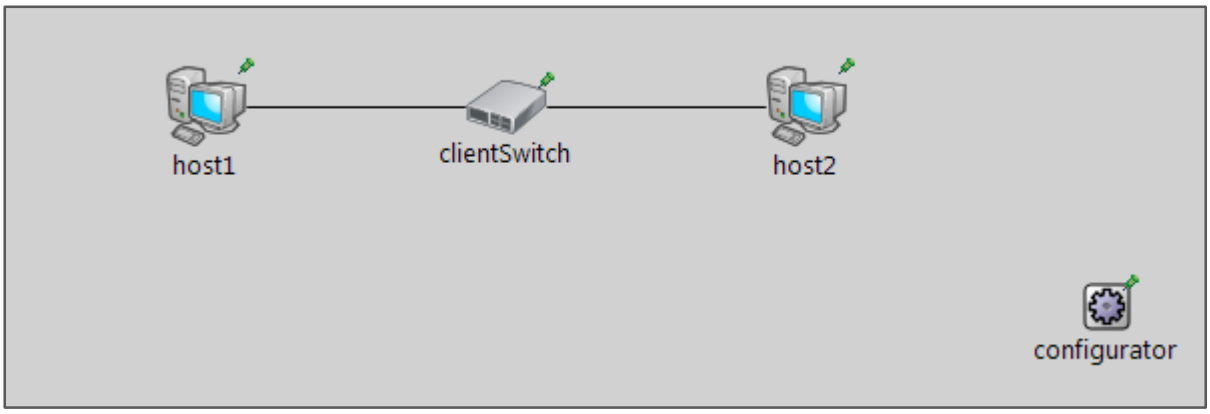
6.3 Laboratorní úloha č.3 – WLAN vs. Ethernet

Cílem laboratorní úlohy je srovnání v současnosti nejvyužívanějších technologií k realizaci počítačových sítí, kabelové spojení technologií Ethernet dle standardu IEEE 802.3 a spojení bezdrátové, využívající k přenosu radiový signál, dle standardu IEEE 802.11.

Úloha srovnává parametry přenosu dat v obou technologiích a zaměřuje se také na přenos řídících dat potřebných k navázání a udržení spojení. Dále se blíže sledují možnosti bezdrátové sítě, například podpora pro řízení kvality služeb a vliv vzdálenosti na chybovost přenášených dat. K simulaci je využito jednoduché zapojení dvou navzájem komunikujících klientů pomocí přístupového bodu, respektive přepínače. Topologii bezdrátové sítě lze vidět na Obr. 15. Zapojení sítě Ethernet lze vidět na Obr. 16. Kompletní návod je uveden v příloze C.



Obr. 15: Topologie bezdrátové sítě



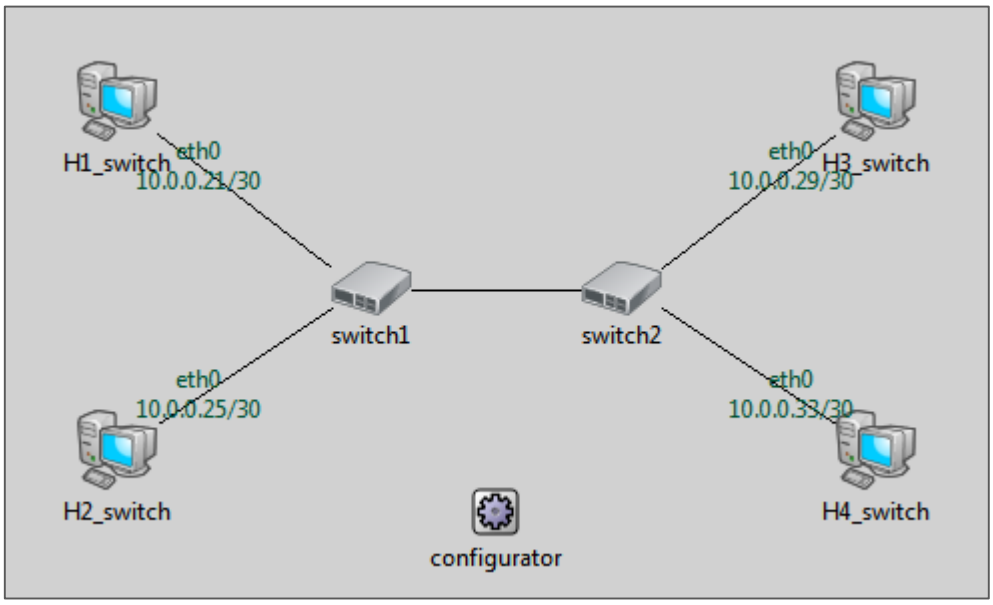
Obr. 16: Topologie kabelové sítě Ethernet

6.4 Laboratorní úloha č.4 – Aktivní síťové prvky

Cílem laboratorní úlohy je srovnání nejpoužívanějších aktivních síťových prvků, které slouží k vytváření počítačových sítí a umožňují komunikaci mezi koncovými stanicemi.

V úloze jsou k propojení sítě použity rozbočovače, přepínače a směrovače, každý z těchto prvků pracuje na jiné vrstvě referenčního modelu ISO/OSI. Tím jsou dány hlavní rozdíly v jejich funkci při zpracování a přeposílání příchozích dat, které lze sledovat ve vytvořené simulaci.

Ke srovnání síťových prvků je sledován parametr zpoždění přenosu souborů službou FTP a množství rozesílaných dat na jednotlivých linkách. Pro simulaci poslouží jednoduchá síť se čtyřmi klienty. Topologii sítě propojené přepínači lze vidět na Obr. 17, ostatní sítě jsou obdobné, pouze je k propojení koncových stanic využito rozbočovačů, respektive směrovačů. Kompletní návod je uveden v příloze D.



Obr. 17: Topologie sítě propojené přepínači

ZÁVĚR

Cílem diplomové práce je návrh čtyř laboratorní úloh, určených k výuce studentů a k rozšíření či ověření jejich znalostí problematiky síťových technologií. K vytvoření těchto úloh bylo využito prostředí síťového simulátoru OMNeT++ s využitím nástavby Inet Framework.

V kapitole 1 je popsána struktura použitých programových prostředků a principy práce s nimi, při vytváření projektů, provádění simulací a následné analýze výsledků.

Další čtyři kapitoly nabízí teoretické podklady pro témata navržených laboratorních úloh. Kapitola 2 je zaměřena na způsoby zajištění kvality síťových služeb, jejich diferenciaci a popis používaných technik pro řízení odesílání paketů. Kapitola 3 popisuje teorii směrovacích protokolů a jejich rozdělení, s bližším zaměřením na protokoly RIPv2 a OSPFv2. Kapitola 4 popisuje technologie pro realizaci lokálních počítačových sítí, z nichž v současnosti nejvyužívanějšími jsou kabelová síť Ethernet popsaná standardem IEEE 802.3 a dalšími modifikacemi tohoto standardu a standard IEEE 802.11 včetně jeho dalších modifikací, které definují sítě bezdrátové s přenosem dat radiovým signálem. K oběma technologiím je uveden stručný technologický popis a definice typických parametrů. Kapitola 5 obsahuje popis základních aktivních síťových prvků, jež se používají k realizaci, zejména lokálních, počítačových sítí, umožňují komunikaci mezi koncovými zařízeními. Popsány jsou zde rozbočovač, přepínač a směrovač, stručné principy jejich funkce a způsobu, jakým zpracovává přijatá data a odesílaná data.

Kapitola 6 obsahuje praktickou část řešení, tedy čtyři vytvořené laboratorní úlohy, Úloha č. 1 - Směrovací protokoly, Úloha č. 2 - Zajištění kvality služeb, Úloha č. 3 – WLAN vs. Ethernet, Úloha č. 4 – Aktivní síťové prvky. Je zde popsán úvod k laboratorní úloze, požadované řešení v rámci vypracování a zapojení topologie sítě určené k simulaci. Všechny úlohy jsou vytvořeny formou kompletního návodu, včetně zdrojového kódu, jehož zkopírováním lze vytvořit testovanou síť i bez znalosti práce s programem OMNeT++. Návod je také průvodcem k vhodnému provedení simulace, následnému zobrazení a analýze získaných výsledků. Kompletní návody ke všem laboratorním úlohám jsou uvedeny v příloze.

LITERATURA

[1] *OMNeT++ User Manual* , 2014. Dostupné online < http://omnetpp.org/documentation>

[2] *OMNeT++ User Guide*, 2014. Dostupné online < http://omnetpp.org/documentation>

[3] *INET Framework for OMNeT++*, 2015. Dostupné online < http://inet.omnetpp.org/Manual.html >

[4] MOLNÁR, K. *Hardware počítačových sítí*. Brno: Vysoké učení technické v Brně, 2012. s. 1-157.

[5] Dostálek, L. *Velký průvodce protokoly TCP/IP a systémem DNS*, Computer Press, Praha 2002.

[6] Halsall, F. *Computer networking and the Internet*. Fourth edition, Addison-Wesley, Edinburg, 2005.

[7] KOTON, J. *Moderní síťové technologie*. Brno: Vysoké učení technické v Brně, 2014. s. 1-191.

[8] Tanenbaum, A. S. *Computer Networks. F*irst edition. Prentice Hall, New Jersey 2003.

[9] MELKA, A., JEŘÁBEK, J. *Komunikační technologie*. Brno: Vysoké učení technické v Brně, 2013. s. 1-172.

[10] FOROUZAN, Behrouz A. *TCP/IP protocolsuite*. Fourth edition, Boston: McGraw-HillHigherEducation, 2010, xxxv, 979 s. ISBN 978-0-07-337604-2.

[11] NOVOTNÝ, V. *Architektura sítí.* Brno: Vysoké učení technické v Brně, 2012. s. 1-152.

[12] Morreale, P. Terplan, K. *The CRC Handbook of Modern Telecommunications*, CRC Press, New York 2001.

[13] *IEEE Standard for Ethernet*, 2012. Dostupné online < http://standards.ieee.org/about/get/802/802.3.html>

[14] BURDA, K. *Návrh, správa a bezpečnost počítačových sítí*. Brno: Vysoké učení technické v Brně, 2014. s. 1-171.

[15] *IEEE Std 802.11*, 2012. Dostupné online < http://standards.ieee.org/about/get/802/802.11.html>

[16] MOLNÁR, K., SOUMAR, M. *Praktikum z informačních sítí*. Brno: Vysoké učení technické v Brně, 2011. s. 1-111.

SEZNAM OBRÁZKŮ

SEZNAM TABULEK

SEZNAM POUŽITÝCH ZKRATEK

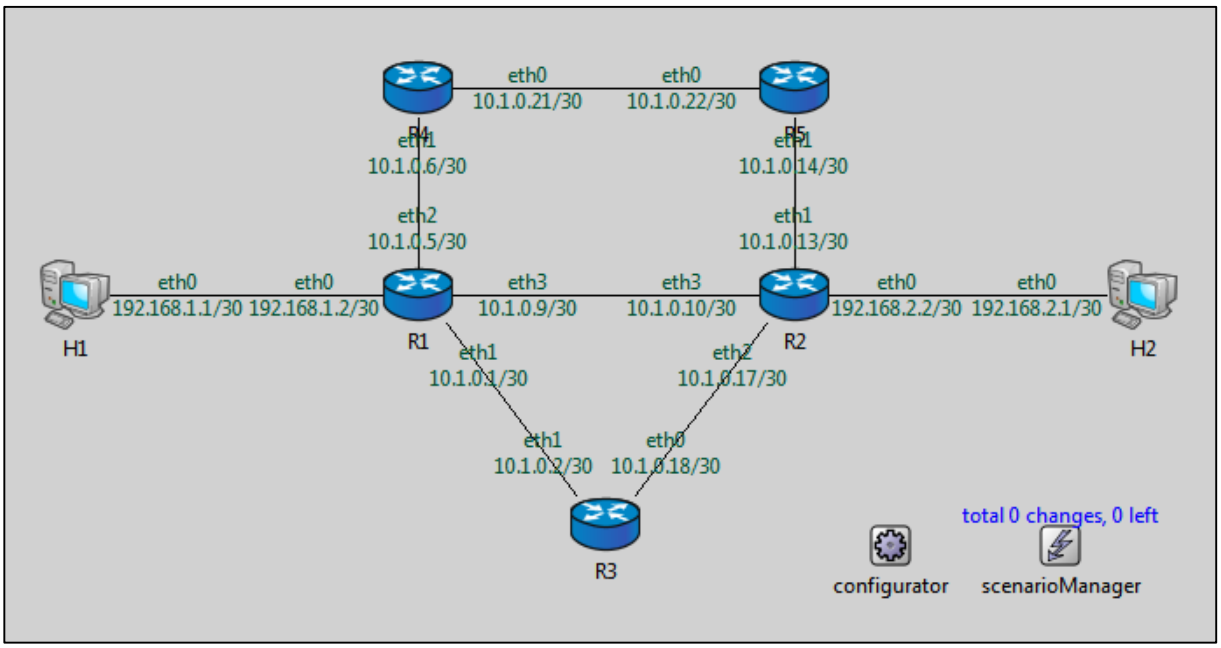
AIFS	Arbitration Interframe Space
ATM	Asynchronous Transfer Mode
BGP	Border Gateway Protocol
CB WFQ	Class-Based Weighted Fair Queuing
CSMA/CA	Carrier Sense Multiple Acces with Collision Avoidance
CSMA/CD	Carrier Sense Multiple Access with Collision Detection
CTS	Clear To Send
CW	Contention Windows
DBD	The Database Description
DIFS	Distributed Interframe Space
DSCP	Differentiated Service Code Point
EDCA	Enhanced Distributed Channel Access
EDCF	Enhanced Distributed Coordination Function
EGP	Exterior Gateway Protocol
EIGRP	Enhanced Interior Gateway Routing Protocol
FC	Frame Control
FEC	Forward Error Correction
FIFO	First In First Out
FQ	Fair Queueing
FTP	File Transfer Protocol
HCCA	HCF Controlled Channel Access
HCF	Hybrid Coordination Function
IDE	Integrated Development Environment
IEEE	Institute of Electrical and Electronics Engineers
IGP	Interior Gateway Protocol
IP	Internet Protocol
IS-IS	Intermediate System to Intermediate System
LAN	Local Area Network
LSA	Link-State Advertisement
LSAck	Link-State Acknowledgement
LSR	Link-State Request
LSU	Link-State Update
MAC	Media Access Control

MAN	Metropolitan Area Network
MIMO	Multiple Input Multiple Output
MPLS	Multiprotocol Label Switching
NED	Network Description
OFDM	Orthogonal Frequency Division Multiplexing
OSPF	Open Shortest Path First
PPP	Point-to-Point Protocol
PQ	Priority Queueing
QoS	Quality of Service
RIP	Routing Information Protocol
RIPng	RIP next generation
RTS	Request To Send
RTT	Round-Trip Time
SC	Sequence Control
SCTP	Stream Control Transmission Protocol
SIFS	Short Interframe Space
SSID	Service Set Identifier
TCP	Transmission Control Protocol
ToS	Type of Service
UDP	User Datagram Protocol
UTP	Unshielded Twisted Pair
VOIP	Voice Over Internet Protocol
WAN	Wide Area Network
WFQ	Weighted Fair Queuing
WLAN	Wireless Local Area Network
WRR	Weighted Round Robin

SEZNAM PŘÍLOH

A. Laboratorní úloha č.1 – Směrovací protokoly

Cílem této laboratorní úlohy je srovnání vlastností směrovacích protokolů RIP a OSPF. Vytvoříte jednoduchou síť s pěti směrovači a dvěma koncovými zařízeními propojenými technologií ethernet. Z výsledků lze sledovat rozdíly ve vytváření směrovacích tabulek, množství dat generovaných směrovači, a to jak při běžném chování sítě, tak i při výpadku spoje mezi směrovači. Pro směrovací protokol OSPF navíc porovnáte vlastnosti sítě s jednou oblastí a sítě rozdělené na dvě menší oblasti. Zapojení topologie sítě lze vidět na obrázku 1.



Obrázek 1: Zapojení topologie sítě

1. Teoretický úvod

Směrovací protokoly slouží pro vytváření směrovacích tabulek popisujících topologii sítě bez nutnosti ručně zapisovat adresy, vyhledávání nejvhodnějších cest podle různých metrik a pravidelně aktualizovat informace všech směrovačů v síti, tak aby byla zaručena konvergence v síti v co možná nejkratším čase. V důsledku stále se rozšiřující globální sítě není možné zajistit konvergenci nasazením jednoho směrovacího protokolu, který by spravoval kompletní síť. Z tohoto důvodu se sítě rozdělí na menší části s různou směrovací politikou. Z pohledu směrování je základním rozdělením protokolů na EGP (Exterior Gateway Protocol), který slouží na globální úrovni k propojování menších lokálních sítí, ve kterých jsou použity protokoly IGP (Interior Gateway Protocol), do této skupiny patří protokoly RIP (router Information Protocol) a OSPF (Open Shortest Path First), na které se tato úloha zaměřuje.

2. Vypracování

Založení nového projektu

Spustíme program **omnetpp.exe**. Pokud se objeví nabídka pro vybrání pracovního adresáře, tento formulář potvrdíme beze změny (vybraná cesta musí být: omnetpp-4.6\samples), jinak vytvořený projekt nebude fungovat.

K vytvoření nového projektu klikneme na **File » New » OMNeT++ Project…** Projekt pojmenujeme VUT loginem, dále stiskneme **Next » Empty project » Finish**. V okně Project Explorer se vytvořila námi pojmenovaná složka, ve které budou uloženy všechny části projektu. Nejprve je zapotřebí přiřadit k projektu odkazy na zdrojové soubory, se kterými se bude dále pracovat. To provedeme kliknutím pravým tlačítkem na vytvořenou složku » **Properties » Project References »** vybereme položku **inet » OK.**

Pro vytvoření projektu jsou zapotřebí dva základní soubory. Network Description File, s příponou .ned, který definuje topologii sítě, použité technologie a síťové prvky. Druhým je Initialization File, přípona .ini, který určuje průběh a parametry simulace.

Klikněte pravým tlačítkem myši na vaši složku a vyberte **New » Network Description File (NED)**, pojmenujte ji **Network.ned » Next » Empty NED File » Finish**.

V souboru NED lze pracovat ve dvou režimech, v grafickém nebo kódovém. Mezi nimi se lze přepínat v levém dolním rohu volbou Design/Source. Jsou navzájem propojené, pokud tedy přidáme prvky sítě z palety, vytvoří se k nim patřičný kód a naopak vložením kódu lze vytvářet celou síť.

Vypracování 1. část - RIP

Nejprve musíme importovat do projektu prvky, se kterými se bude pracovat. Do souboru Network.ned v režimu Source vložíme následující kód:

```
import inet.common.lifecycle.LifecycleController;
import inet.common.misc.ThruputMeteringChannel;
import inet.common.scenario.ScenarioManager;
import inet.networklayer.configurator.ipv4.IPv4NetworkConfigurator;
import inet.networklayer.ipv4.IPv4RoutingTable;
import inet.networklayer.ipv4.RoutingTableRecorder;
import inet.node.inet.StandardHost;
import inet.node.ospfv2.OSPFRouter;
import inet.node.rip.RIPRouter;
```

Dále vytvoříme síť pojmenovanou RIP, ta bude obsahovat 5 routerů nastavených pro směrovací protokol RIP a 2 koncová zařízení. Prvky sítě se definují v sekci submodules, kde se zapíše jméno a požadovaný prvek, v tomto případě použijeme StandardHost pro koncová zařízení a RIPRouter pro směrovače. Prvkům lze přiřazovat různé parametry, zatím nám stačí

pouze nadefinovat polohu zobrazení v grafické topologii a počet, respektive typ aktivních rozhraní.

Vytvoříme 2 přenosové kanály a definujeme jejich parametry. Všechny prvky v síti jsou propojeny technologií ethernet s přenosovou rychlostí 100Mbit/s respektive 10Mbit/s a oba se zpožděním 0,1µs. Zvolíme typ ThruputMeteringChannel.

```
network RIP
{   parameters:
        @display("p=10,10;b=712,152;bgb=661,348");
    types:
        channel LAN100 extends ThruputMeteringChannel
        {   delay = 0.1us;
            datarate = 100Mbps;
        }
      channel LAN10 extends ThruputMeteringChannel
        {   delay = 0.1us;
            datarate = 10Mbps;
        }
    submodules:
        H1: StandardHost {
            parameters:                 @display("p=42,154");
            gates:                  ethg[1];
        }
        H2: StandardHost {
            parameters:                 @display("p=607,154");
            gates:                  ethg[1];
        }
        R1: RIPRouter {
            parameters:                 @display("p=216,154");
            gates:                  ethg[4];
        }
        R2: RIPRouter {
            parameters:                 @display("p=431,154");
            gates:                  ethg[4];
        }
        R3: RIPRouter {
            parameters:                 @display("p=323,285");
            gates:                  ethg[2];
        }
        R4: RIPRouter {
            parameters:                 @display("p=216,36");
            gates:                  ethg[2];
        }
        R5: RIPRouter {
            parameters:                 @display("p=431,36");
            gates:                  ethg[2];
        }
      scenarioManager: ScenarioManager {@display("p=599,297");
        }
```

Dále nakonfigurujeme síťovou adresaci, k tomu využijeme prvek IPv4NetworkConfigurator s následujícími parametry. Pro připojení koncových zařízení k nejbližšími směrovači nastavíme privátní IP adresy z rozsahu třídy C a pro spoje mezi všemi routery zvolíme adresy z rozsahu třídy A privátních adres. Také nastavíme výchozí brány pro obě koncová zařízení a povolíme multicastovou komunikaci na všech aktivních rozhraní směrovačů, ten je využíván pro rozesílání směrovacích informací.

```
configurator: IPv4NetworkConfigurator {
   parameters:
                @display("p=502,297");
                config = xml("<config>"
                        + "<interface among='H1 R1'
address='192.168.1.x' netmask='255.255.255.x' />"
                        + "<interface among='H2 R2'
address='192.168.2.x' netmask='255.255.255.x' />"

                        + "<interface among='R*' address='10.1.0.x'
netmask='255.255.255.x' />"

                        + "<route hosts='H1' destination='*'
gateway='R1'/>"
                        + "<route hosts='H2' destination='*'
gateway='R2'/>"

                        + "<route hosts='R*' destination='224.0.0.0'
netmask='240.0.0.0' interface='eth0'/>"
                        + "<route hosts='R*' destination='224.0.0.0'
netmask='240.0.0.0' interface='eth1'/>"
                        + "<route hosts='R1 R2' destination='224.0.0.0'
netmask='240.0.0.0' interface='eth2'/>"
                        + "<route hosts='R1 R2' destination='224.0.0.0'
netmask='240.0.0.0' interface='eth3'/>"

                        + "</config>");         }
```

K dokončení návrhu už zbývá pouze propojit prvky sítě. Jedná se o plně duplexní spojení s parametry kanálů LAN100 a LAN10, které jsme nadefinovali již v předchozích krocích. Věnujte pozornost přenosovým rychlostem přiřazených jednotlivým linkám, z tohoto nastavení lze následně pozorovat metodiku směrování v síti.

```
    connections:
        H1.ethg[0] <--> LAN100 <--> R1.ethg[0];
        H2.ethg[0] <--> LAN100 <--> R2.ethg[0];
        R5.ethg[1] <--> LAN100 <--> R2.ethg[1];
        R4.ethg[0] <--> LAN100 <--> R5.ethg[0];
        R1.ethg[2] <--> LAN100 <--> R4.ethg[1];
        R1.ethg[3] <--> LAN100 <--> R2.ethg[3];
        R1.ethg[1] <--> LAN10 <--> R3.ethg[1];
        R2.ethg[2] <--> LAN10 <--> R3.ethg[0];
}
```

Tímto je vytvořená síť kompletně zprovozněna k provádění simulací, nyní je zapotřebí určit parametry a průběh simulace. Jelikož soubor NED je statický, je vhodné v něm definovat pouze základní parametry, které nechceme dále měnit ani nijak upravovat. Pro nastavení průběhu simulace a síťového provozu slouží konfigurační soubor s příponou .ini. Spuštěním tohoto souboru zahájíme simulaci.

Do složky s projektem vytvoříme nový soubor, **New » Initialization File (ini) »** můžete ponechat název **omnetpp.ini » Next » Empty Ini File » Finish**.

Pro potřeby simulace je možné v jednom souboru vytvářet několik různých konfigurací, k vybrání požadované budeme vyzváni po spuštění simulace. Pro vytvoření nové konfigurace se používá syntaxe [Config nazev_konfigurace] zapsaná v hranatých závorkách. Hlavní částí je sekce [General], která obsahuje nastavení společná pro všechny vytvořené konfigurace, jejím nastavením začneme.

Vložíme následujícíkód, který povolí výpis problému při chybně nastavené simulaci, nastaví se cesta k potřebným doplňkům a omezí se maximální trvání simulace na 10 minut.

```
debug-on-errors = true
tkenv-plugin-path = ../inet/etc/plugins
sim-time-limit = 600s
```

Dále zakážeme vytváření statických směrovacích tabulek:

```
*.configurator.addStaticRoutes = false
*.configurator.addSubnetRoutes = false
*.configurator.addDefaultRoutes = false
```

Dále vytvoříme novou konfiguraci, **[Config RIP]**, kterou vložíme na konec kódu souboru **omnetpp.ini** a do ní zapíšeme kód, kterým specifikujeme použitou síť a cestu k souboru s konfigurací metriky v síti:

```
network = RIP
**.rip.ripConfig = xmldoc("RIP.xml")
```

Při pojmenovávání konfigurací se snažíme volit výstižné názvy, jelikož budeme vytvářet vícero konfigurací, usnadní se tímto orientace v jednotlivých simulacích a v souborech z výsledky, které se pojmenují dle názvu zvolené konfigurace.

Tento soubor nyní musíme vytvořit, **New » File » RIP.xml » Finish**, a vložením následujícího kódu nastavíme metriku pro všechny spoje v síti na hodnotu 1.

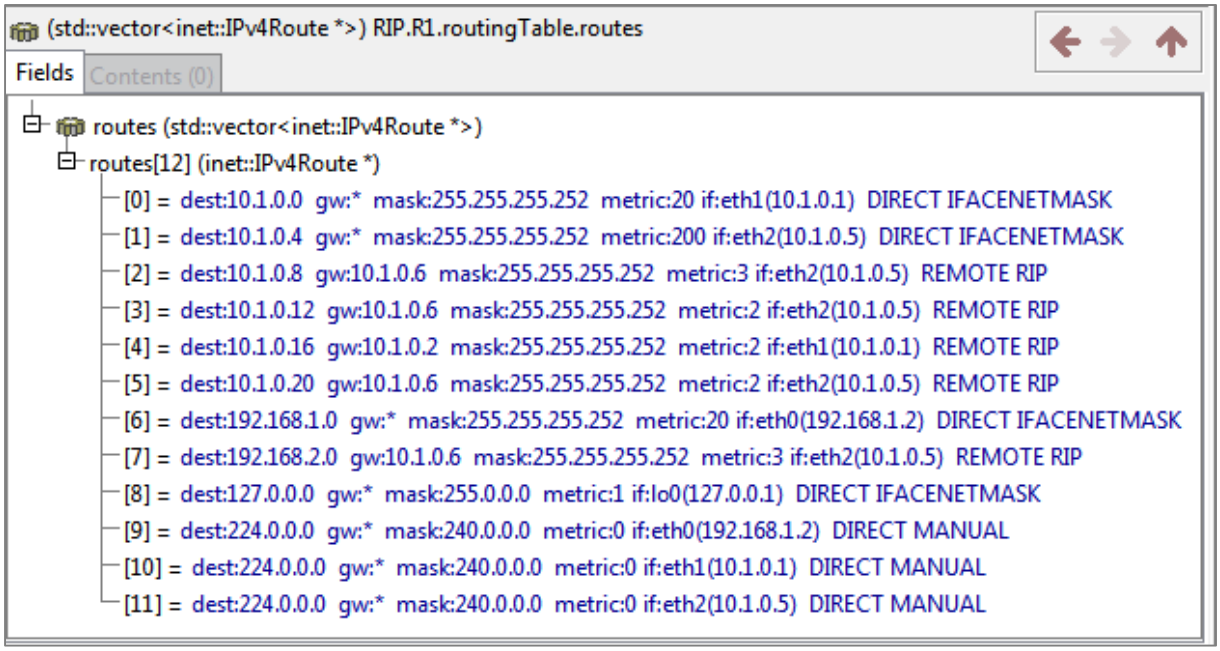
```
<?xml version="1.0"?>
<RIPConfig>
  <interface among="R? R?" metric="1"/>
</RIPConfig>
```

Tato hodnota odpovídá počtu přeskoků při průchodu dat směrovači, což je jediná metoda metriky implementovaná směrovacím protokolem RIP. Z toho také vyplývá hlavní výhoda při použití RIP a to jednoduchost implementace. Tímto krokem je konfigurace směrování kompletně dokončena. Vytvoření tohoto souboru navíc není povinné, pokud nepotřebujeme měnit počet přeskoků při průchodu rozhraním, jelikož hodnota metriky je defaultně nastavena na 1, ale pro účely dalšího testování lze tento zápis rozšířit. Syntaxe je patrná z výše uvedeného kódu, kdy se definuje nastavovaný spoj a následně jeho metrika, otazník nahrazuje libovolný znak.

Před prvním spuštěním simulace je potřeba sestavit projekt, klikneme pravým tlačítkem myši na složku s projektem **» Build Configurations » Build Selected… »** zaškrtneme **gcc-debug » OK**.

Po dokončení operace můžeme spustit simulaci, klikneme pravým tlačítkem myši na soubor **RIP.ini » Run As » OMNeT++ Simulation »** vybereme konfiguraci pojmenovanou **RIP » OK.**

Pro zobrazení směrovací tabulky klikneme myší na jeden ze směrovačů a v okně s obsahem (obvykle v levém dolním rohu) vybereme **routingTable » routes**.

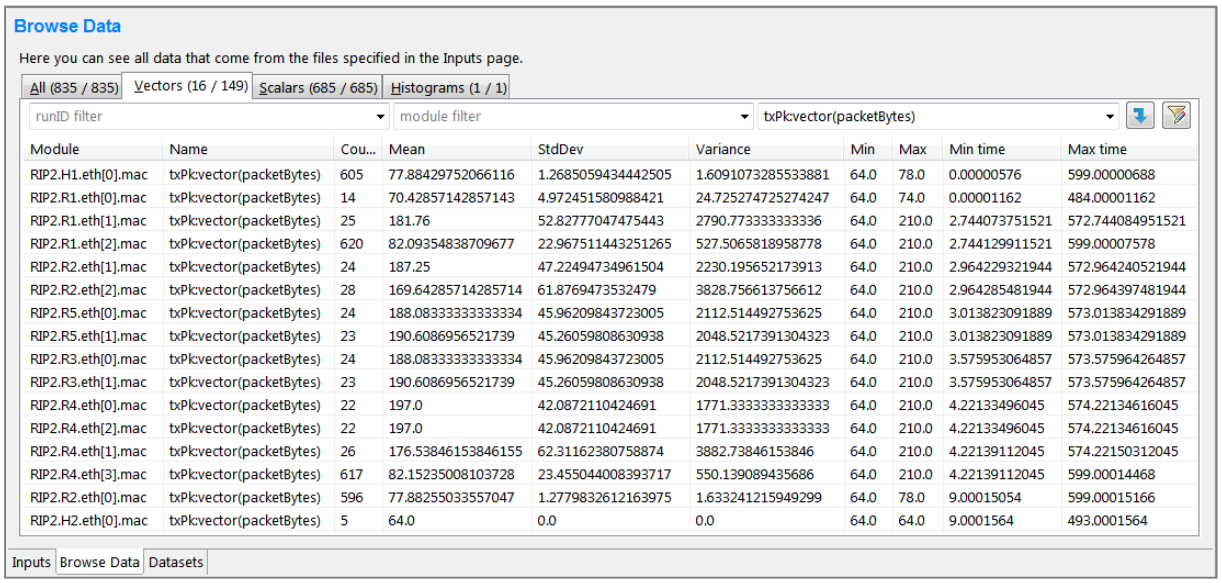


Obrázek 2: Zobrazení směrovací tabulky protokolu RIP směrovače R1

Stisknutím ikony Run (nebo klávesou F5) spustíme simulaci, můžeme sledovat výpis poslaných zpráv, které můžeme vidět také v animaci. Také se zaměříme na proces vytváření záznamů ve směrovací tabulce. Pro rychlejší dokončení můžeme stisknout ikonu Express (F7).

Po dokončení simulace se vytvoří soubory s výsledky ve složce **results**. Dvojklikem otevřeme jeden z nich a potvrdíme vytvoření souboru RIP.anf. V souborech s příponou .anf můžeme prohlížet výsledky provedených simulací, v záložce **Browse Data** máme na výběr z výsledků zobrazených vektorově v grafu a skalárně.

Pro zobrazení provozu směrovačů si otevřeme záložku **Vectors »** v poli statistic name filter vybereme **txPk:vector(packetBytes)**. Zobrazí se nám přenos dat všech aktivních rozhraní na směrovačích. Dvojklikem vykreslíme graf, na kterém můžeme vidět postupné budování a rozšiřování směrovací tabulky a následné rozesílání v pravidelných intervalech.



Browse Data

Here you can see all data that come from the files specified in the Inputs page.

All (835 / 835) | Vectors (16 / 149) | Scalars (685 / 685) | Histograms (1 / 1)

runID filter | module filter | txPk:vector(packetBytes)

Module	Name	Cou...	Mean	StdDev	Variance	Min	Max	Min time	Max time
RIP2.H1.eth[0].mac	txPk:vector(packetBytes)	605	77.88429752066116	1.2685059434442505	1.6091073285533881	64.0	78.0	0.00000576	599.00000688
RIP2.R1.eth[0].mac	txPk:vector(packetBytes)	14	70.42857142857143	4.972451580988421	24.725274725274247	64.0	74.0	0.00001162	484.00001162
RIP2.R1.eth[1].mac	txPk:vector(packetBytes)	25	181.76	52.82777047475443	2790.773333333336	64.0	210.0	2.744073751521	572.744084951521
RIP2.R1.eth[2].mac	txPk:vector(packetBytes)	620	82.09354838709677	22.967511443251265	527.5065818958778	64.0	210.0	2.744129911521	599.00007578
RIP2.R2.eth[1].mac	txPk:vector(packetBytes)	24	187.25	47.22494734961504	2230.195652173913	64.0	210.0	2.964229321944	572.964240521944
RIP2.R2.eth[2].mac	txPk:vector(packetBytes)	28	169.64285714285714	61.8769473532479	3828.756613756612	64.0	210.0	2.964285481944	572.964397481944
RIP2.R5.eth[0].mac	txPk:vector(packetBytes)	24	188.08333333333334	45.96209843723005	2112.514492753625	64.0	210.0	3.013823091889	573.013834291889
RIP2.R5.eth[1].mac	txPk:vector(packetBytes)	23	190.6086956521739	45.26059808630938	2048.5217391304323	64.0	210.0	3.013823091889	573.013834291889
RIP2.R3.eth[0].mac	txPk:vector(packetBytes)	24	188.08333333333334	45.96209843723005	2112.514492753625	64.0	210.0	3.575953064857	573.575964264857
RIP2.R3.eth[1].mac	txPk:vector(packetBytes)	23	190.6086956521739	45.26059808630938	2048.5217391304323	64.0	210.0	3.575953064857	573.575964264857
RIP2.R4.eth[0].mac	txPk:vector(packetBytes)	22	197.0	42.0872110424691	1771.3333333333333	64.0	210.0	4.22133496045	574.22134616045
RIP2.R4.eth[2].mac	txPk:vector(packetBytes)	22	197.0	42.0872110424691	1771.3333333333333	64.0	210.0	4.22133496045	574.22134616045
RIP2.R4.eth[1].mac	txPk:vector(packetBytes)	26	176.53846153846155	62.31162380758874	3882.73846153846	64.0	210.0	4.22139112045	574.22150312045
RIP2.R4.eth[3].mac	txPk:vector(packetBytes)	617	82.15235008103728	23.455044008393717	550.139089435686	64.0	210.0	4.22139112045	599.00014468
RIP2.R2.eth[0].mac	txPk:vector(packetBytes)	596	77.88255033557047	1.2779832612163975	1.633241215949299	64.0	78.0	9.00015054	599.00015166
RIP2.H2.eth[0].mac	txPk:vector(packetBytes)	5	64.0	0.0	0.0	64.0	64.0	9.0001564	493.0001564

Inputs | Browse Data | Datasets

Obrázek 3: Zobrazení výsledků ve vektorové podobě

Vypracování 2. část – OSPF

Vytvoříme síť se stejnou topologií, ale nyní pro směrování použijeme protokol OSPF. Porovnáme směrovací tabulky a síťový provoz z výsledky předchozí simulace.

Pro vytvoření sítě se směrovacím protokolem otevřeme soubor **Network.ned** a zkopírujeme celý kód popisující síť RIP. Ve zkopírované části změníme název sítě na **network OSPF** a přejmenujeme také všechny směrovače na **OSPFRouter.**

Upravíme nastavení prvku IPv4NetworkConfigurator přidáním tohoto řádku:

```
+ "<multicast-group hosts='R*' address='224.0.0.5 224.0.0.6' />"
```

V souboru omnetpp.ini vytvoříme novou konfiguraci **[Config OSPF]** a do ní vložíme následující kód:

```
network = OSPF
**.ospf.ospfConfig = xmldoc("OSPF.xml")
```

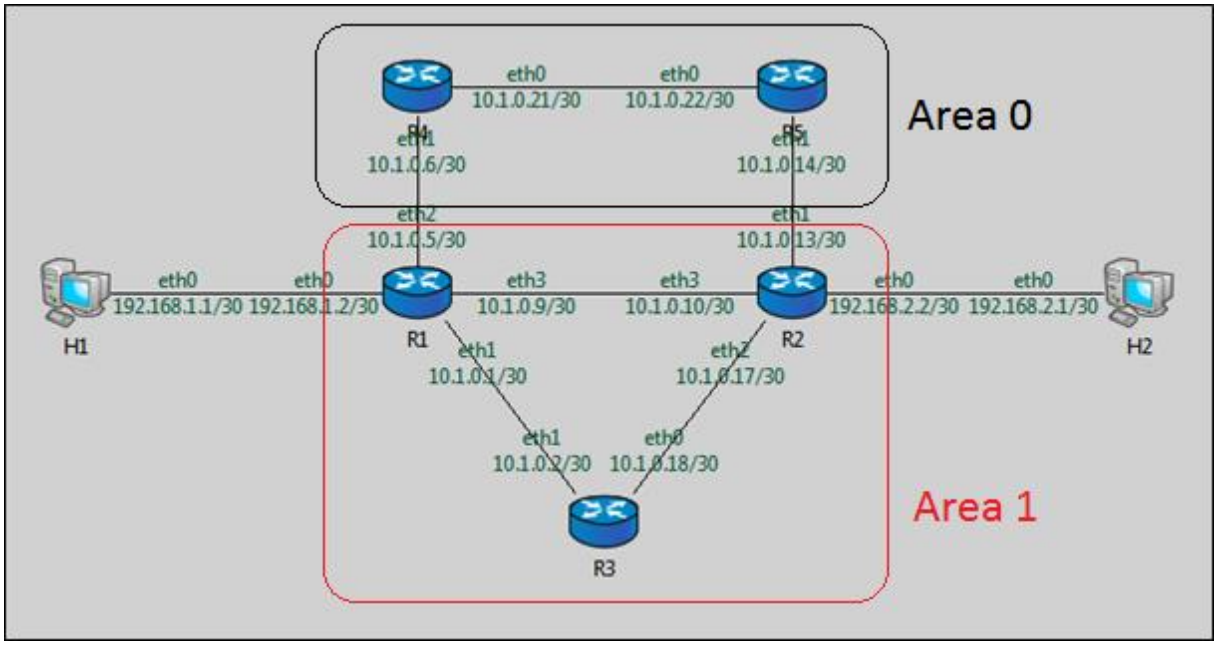
Vytvoříme nový soubor s názvem **OSPF.xml,** ve kterém lze rozdělit síť na více OSPF oblastí a nastavit metriku jednotlivým linkám.

```
<?xml version="1.0"?>
<OSPFASConfig xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="OSPF.xsd">

  <!-- Areas -->
  <Area id="0.0.0.0">

  </Area>

  <!-- Routers -->
  <Router name="R1" RFC1583Compatible="true">
    <BroadcastInterface ifName="eth0" areaID="0.0.0.0"
interfaceOutputCost="10" />
    <PointToPointInterface ifName="eth1" areaID="0.0.0.0"
interfaceOutputCost="100" />
    <PointToPointInterface ifName="eth2" areaID="0.0.0.0"
interfaceOutputCost="10" />
    <PointToPointInterface ifName="eth3" areaID="0.0.0.0"
interfaceOutputCost="10" />
  </Router>

  <Router name="R2" RFC1583Compatible="true">
    <BroadcastInterface ifName="eth0" areaID="0.0.0.0"
interfaceOutputCost="10" />
    <PointToPointInterface ifName="eth1" areaID="0.0.0.0"
interfaceOutputCost="10" />
    <PointToPointInterface ifName="eth2" areaID="0.0.0.0"
interfaceOutputCost="100" />
    <PointToPointInterface ifName="eth3" areaID="0.0.0.0"
interfaceOutputCost="10" />
  </Router>

  <Router name="R3" RFC1583Compatible="true">
    <PointToPointInterface ifName="eth0" areaID="0.0.0.0"
interfaceOutputCost="100" />
    <PointToPointInterface ifName="eth1" areaID="0.0.0.0"
interfaceOutputCost="100" />
  </Router>

  <Router name="R4" RFC1583Compatible="true">
    <PointToPointInterface ifName="eth0" areaID="0.0.0.0"
interfaceOutputCost="10" />
    <PointToPointInterface ifName="eth1" areaID="0.0.0.0"
interfaceOutputCost="10" />
  </Router>

  <Router name="R5" RFC1583Compatible="true">
    <PointToPointInterface ifName="eth0" areaID="0.0.0.0"
interfaceOutputCost="10" />
    <PointToPointInterface ifName="eth1" areaID="0.0.0.0"
interfaceOutputCost="10" />
  </Router>
</OSPFASConfig>
```

Nyní můžete spustit simulaci, porovnejte výsledky s poznatky, které jste získali sledováním protokolu RIP. Dále zobrazte graf přenášených dat na vybraném rozhraní, můžete na něm vidět periodické rozesílání hello paketů a vyšší aktivitu při inicializaci sítě, kdy směrovače shromažďují a navzájem si vyměňují informace o kompletní topologii sítě. Můžete vidět výrazně menší velikost hello zpráv oproti protokolu RIP, který rozesílá kompletní směrovací tabulku na všechna aktivní rozhraní směrovače.

Vypracování 3. část – OSPF s rozdělením na oblasti

Upravíme stávající síť rozdělením směrovačů do dvou oblastí, viz Obrázek 4.



Obrázek 4: Topologie s rozdělením na oblasti

V souboru omnetpp.ini vytvoříme novou konfiguraci vložením kódu: **[Config OSPF_areas]**, a do ní zapíšeme cestu k novému konfigurační souboru s názvem OSPF_areas.xml:

```
extends = OSPF
**.ospf.ospfConfig = xmldoc("OSPF_areas.xml")
```

Tento soubor nyní vytvoříme a vložíme do něj následující kód. V něm jsou specifikovány dvě oblasti a nastavení jednotlivých rozhraní všech směrovačů. Kromě přiřazení rozhraní do požadované oblasti, lze nastavit i metriku linek.

```
<?xml version="1.0"?>
<OSPFASConfig xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="OSPF.xsd">

  <!-- Areas -->
  <Area id="0.0.0.0">
  </Area>
  <Area id="0.0.0.1">
  </Area>

<!-- Routers -->
  <Router name="R1" RFC1583Compatible="true">
    <BroadcastInterface ifName="eth0" areaID="0.0.0.1"
interfaceOutputCost="10" />
    <PointToPointInterface ifName="eth1" areaID="0.0.0.1"
interfaceOutputCost="100" />
    <PointToPointInterface ifName="eth2" areaID="0.0.0.0"
interfaceOutputCost="10" />
    <PointToPointInterface ifName="eth3" areaID="0.0.0.1"
interfaceOutputCost="10" />
  </Router>

  <Router name="R2" RFC1583Compatible="true">
    <BroadcastInterface ifName="eth0" areaID="0.0.0.1"
interfaceOutputCost="10" />
    <PointToPointInterface ifName="eth1" areaID="0.0.0.0"
interfaceOutputCost="10" />
    <PointToPointInterface ifName="eth2" areaID="0.0.0.1"
interfaceOutputCost="100" />
    <PointToPointInterface ifName="eth3" areaID="0.0.0.1"
interfaceOutputCost="10" />
  </Router>

  <Router name="R3" RFC1583Compatible="true">
    <PointToPointInterface ifName="eth0" areaID="0.0.0.1"
interfaceOutputCost="100" />
    <PointToPointInterface ifName="eth1" areaID="0.0.0.1"
interfaceOutputCost="100" />
  </Router>

  <Router name="R4" RFC1583Compatible="true">
    <PointToPointInterface ifName="eth0" areaID="0.0.0.0"
interfaceOutputCost="10" />
    <PointToPointInterface ifName="eth1" areaID="0.0.0.1"
interfaceOutputCost="10" />
  </Router>

  <Router name="R5" RFC1583Compatible="true">
    <PointToPointInterface ifName="eth0" areaID="0.0.0.0"
interfaceOutputCost="10" />
    <PointToPointInterface ifName="eth1" areaID="0.0.0.1"
interfaceOutputCost="10" />
  </Router>

</OSPFASConfig>
```

Spusťte simulaci a porovnejte, jak se změní výsledné směrovací tabulky oproti OSPF síti bez rozdělení do oblastí.

Vypracování 4. část – Cesta sítí RIP vs. OSPF

V obou vytvořených sítích nastavíme jednosměrný datový tok mezi koncovými stanicemi, na kterém lze sledovat zvolenou cestu v závislosti na typu a zvoleném nastavení použitého směrovacího protokolu.

Do sekce General v konfiguračním souboru vložíme následující kód:

```
**.numUdpApps = 1
**.H1.udpApp[0].typename = "UDPBasicApp"
**.H1.udpApp[0].destPort = 1234
**.H1.udpApp[0].messageLength = 32 bytes
**.H1.udpApp[0].sendInterval = 1s
**.H1.udpApp[0].startTime = 0s
**.H1.udpApp[0].stopTime = this.startTime + 600s
**.H1.udpApp[0].destAddresses = "H2"

**.H2.udpApp[0].typename = "UDPSink"
**.H2.udpApp[0].localPort = 1234
```

Tím zprovozníme jednoduchý datový tok, který vysílá pakety s intervalem jedné sekundy od stanice H1 ke stanici H2, pro lepší přehlednost jsou data přenášena pouze v tomto směru a bez potvrzování přijatých paketů (protokol UDP).

Dále vytvoříme novou konfiguraci **[Config RIP_vypadek_linky]** a vložíme do ní odkaz na soubor se scénářem výpadku linky, Vypadek_linky.xml:

```
extends = RIP
*.scenarioManager.script = xmldoc("Vypadek_linky.xml")
```

Vytvoříme obdobnou konfiguraci **[Config OSPF_vypadek_linky]**, s následujícím kódem:

```
extends = OSPF
*.scenarioManager.script = xmldoc("Vypadek_linky.xml")
```

Vytvoříme soubor s názvem **Vypadek_linky.xml** a nastavíme v něm výpadek linky mezi směrovači R1 a R2 v čase 200 sekund a opětovné obnovení spoje v čase 500 sekund.

```
<scenario>
    <at t="200">
        <disconnect src-module="R2" src-gate="ethg$o[3]" />
        <disconnect src-module="R1" src-gate="ethg$o[3]" />
    </at>
    <at t="500">
        <connect src-module="R1" src-gate="ethg[3]"
                 dest-module="R2" dest-gate="ethg[3]"
                 channel-type="inet.common.misc.ThruputMeteringChannel">
            <param name="delay" value="0.1us" />
            <param name="datarate" value="100Mbps" />
        </connect>
    </at>
</scenario>
```

Spustíme oba nově vytvořené scénáře s výpadkem linky a porovnáme je mezi sebou. Koncová stanice H1 se snaží vysílat data ihned v čase 0, ale nejprve musí směrovače shromáždit informace o topologii sítě. V okně s výpisem zpráv spuštěné simulace lze vidět, v jakém čase se začnou doručovat první UDP pakety, neboli jak dlouho trvá sestavení směrovacích tabulek.

Dále porovnáme volbu cesty sítí pro oba protokoly. V otevřené simulaci klikneme pravým tlačítkem myši na grafické zobrazení topologie **» Animation Filter… »** záložka **Filtering »** a zadáme následující podmínku:

```
not UDPBasicAppData-* and className(inet::EthernetIIFrame)
```

Tímto vyfiltrujeme všechna nechtěná data a v animaci se budou zobrazovat pouze pakety typu UDP pro snazší viditelnost směrování.

Pro urychlení simulace můžete stisknout ikonu **UNTIL (Ctrl+F5) »** Simulation time to stop at: **200s » OK » EXPRESS (F7)**. Tím se v simulaci okamžitě posunete do času výpadku linky, po opětovném spuštění simulace normálním rychlostí můžete pozorovat, jak protokoly reagují na změněnou topologii. Pozorujte, jakou cestu volí protokol RIP a jakou OSPF po výpadku linky a co způsobuje odlišné chování v simulované síti.

Potom se můžete přesunout na čas **500s,** opět pozorujte jakým způsobem a v jakém čase se jednotlivé směrovací protokoly reagují na změnu.

Vypracování 5. část – Cesta sítí OSPF s rozdělením na oblasti

Do scénáře **[Config OSPF_oblasti]** přidáme naplánovaný výpadek linky:

```
*.scenarioManager.script = xmldoc("Vypadek_linky.xml")
```

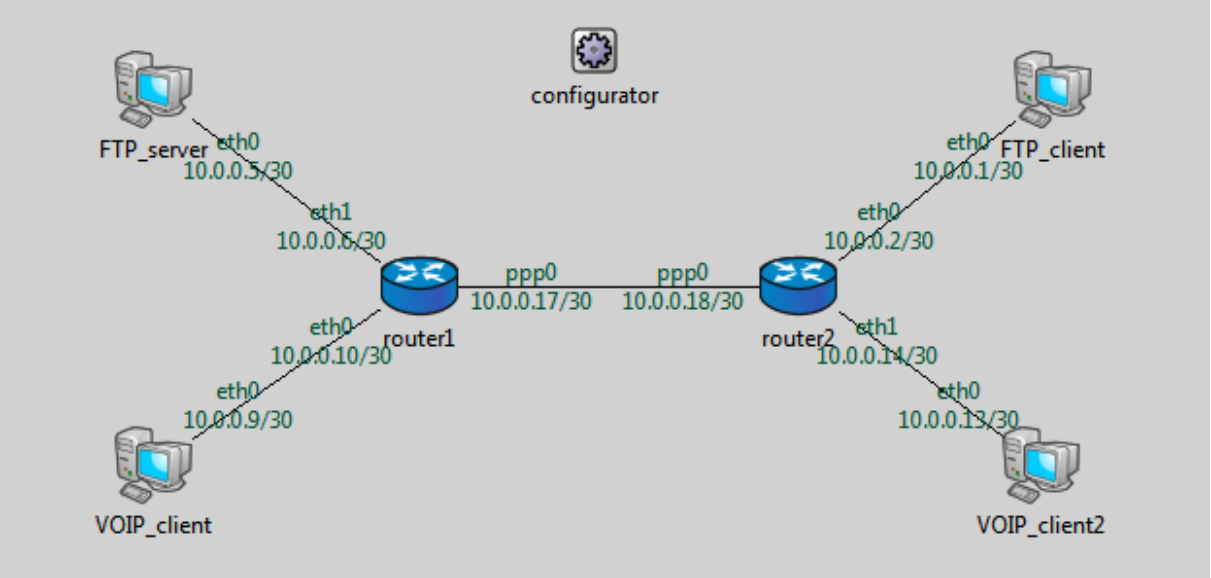
Spustíme tento scénář a po proběhnutí simulace do času výpadku linky (200s) můžeme vidět, že pakety jsou směrovány jinou trasou než v případě sítě s jedinou oblastí, jelikož směrovače preferují cestu bez přecházení do jiných oblastí

Otázky:

1. Jak často posílají směrovače s protokolem RIP aktualizace a co je jejich obsahem?
2. Jak se změní obsah směrovací tabulky směrovačů s OSPF při rozdělení sítě na menší oblasti?
3. Co je rozhodujícím faktorem pro volbu cesty sítí pro RIP vs. OSPF?

B. Laboratorní úloha č.2 – Zajištění kvality služeb

Cílem této laboratorní úlohy je srovnání různých mechanizmů řízeného odesílání paketů a jejich vlivu na parametry definující kvalitu síťového přenosu (zpoždění, proměnlivost zpoždění, ztrátovost paketů). K simulaci poslouží jednoduchá síť s linkou poskytující nedostatečnou šířku pásma pro definovaný provoz a koncové stanice současně generující FTP datový přenos a VOIP hovor. Zapojení topologie sítě lze vidět na obrázku 1.



Obrázek 1: Zapojení topologie sítě

1. Teoretický úvod

Vzhledem ke stále se zvětšujícímu množství síťových služeb s odlišnými požadavky na parametry přenosu, hrozí riziko vysokého vytížení komunikačních linek a bez řízení úrovně kvality by bylo velmi omezeno využití určitých služeb, zejména těch pracujících v reálném čase, jejichž hlavním požadavkem je nízké časové zpoždění a jeho kolísání (jitter). Naopak pro většinu datových služeb je hlavním požadavkem co nejnižší ztrátovost či chybovost paketů a propustnost. Proto bylo nutné zavedení mechanizmů rozlišování typů služeb podle jejich typických požadavků a zajištění odlišných úrovní kvality služeb.

Pro zajištění kvality služeb je základním požadavkem rozlišení toku dat do tříd, podle typu dat zdrojových aplikací. Poté je pro každou třídu definováno odlišné zacházení, mechanizmus pro zajištění kvality služeb je označován pojmem QoS – Quality of Service. Ten také zajišťuje dohled nad síťovým provozem, aktivní správu front jednotlivých tříd, plánuje odesílání paketů a upravuje provoz pro lepší využití dostupné kapacity komunikační linky. Základním mechanizmem je fronta FIFO, která neposkytuje řízení odesílání, tzv. služba „Best-Effort“, zástupci pokročilejších systému jsou PQ (prioritní systém front) nebo WFQ (systém front s váženou spravedlivou obsluhou). Tyto tři mechanizmy porovnáte v laboratorní úloze.

2. Vypracování

Založení nového projektu

Spustíme program **omnetpp.exe**. Pokud se objeví nabídka pro vybrání pracovního adresáře, tento formulář potvrdíme beze změny (vybraná cesta musí být: omnetpp-4.6\samples), jinak vytvořený projekt nebude fungovat.

K vytvoření nového projektu klikneme na **File » New » OMNeT++ Project…** Projekt pojmenujeme VUT loginem, dále stiskneme **Next » Empty project » Finish**. V okně Project Explorer se vytvořila námi pojmenovaná složka, ve které budou uloženy všechny části projektu. Nejprve je zapotřebí přiřadit k projektu odkazy na zdrojové soubory, se kterými se bude dále pracovat. To provedeme kliknutím pravým tlačítkem na vytvořenou složku » **Properties » Project References »** vybereme položku **inet » OK**.

Pro vytvoření projektu jsou zapotřebí dva základní soubory. Network Description File, s příponou .ned, který definuje topologii sítě, použité technologie a síťové prvky. Druhým je Initialization File, přípona .ini, který určuje průběh a parametry simulace.

Klikněte pravým tlačítkem myši na vaši složku a vyberte **New » Network Description File (NED)**, pojmenujte ji **Network.ned » Next » Empty NED File » Finish**.

V souboru NED lze pracovat ve dvou režimech, v grafickém nebo kódovém. Mezi nimi se lze přepínat v levém dolním rohu volbou Design/Source. Jsou navzájem propojené, pokud tedy přidáme prvky sítě z palety, vytvoří se k nim patřičný kód a naopak vložením kódu lze vytvářet celou síť.

Vypracování 1. část - FIFO

Nejprve musíme importovat do projektu prvky, se kterými se bude pracovat. Do souboru Network.ned vložíme následující kód:

```
import inet.networklayer.configurator.ipv4.IPv4NetworkConfigurator;
import inet.node.inet.Router;
import inet.node.inet.StandardHost;
import inet.common.misc.ThruputMeteringChannel;
```

Vytvoříme síť pojmenovanou DiffservNetwork, složenou ze čtyř prvků StandardHost, které poslouží jako stanice generující provoz v síti. Nazveme je FTP_client, FTP_server a VOIP_client1, VOIP_client2. Budeme tedy sledovat diferencování prostředků služeb pro FTP přenos a pro VOIP hovor. Spojení mezi stanicemi je realizováno dvěma směrovači, propojenými technologií PPP s nedostačující přenosovou rychlostí. Na PPP portu routeru 1 tedy bude docházet k nadměrným požadavkům na zatížení sítě a bude na něm možné sledovat vlastnosti různých systému pro řízení kvality služeb.

Nyní vytvoříme přenosové kanály a definujeme jejich parametry. Všechny stanice v síti jsou připojeny ke směrovačům technologií ethernet s přenosovou rychlostí 10Mbit/s a se zpožděním 0,1µs. Pro spojení mezi směrovači vytvoříme kanál PPP s přenosovou rychlostí 1Mbit/s. Pro vytvoření kanálů zvolíme typ ThruputMeteringChannel, který umožňuje zobrazení požadovaného parametru přímo v grafickém režimu simulace. Nastavíme ho na zobrazení počtu přenesených paketů.

```
network DiffservNetwork
{
    parameters:
        @display("bgb=644,329");
    types:
        channel PPP extends ThruputMeteringChannel
        {
            delay = normal(0.004s, 0.0018s);
            datarate = 1Mbps;
            thruputDisplayFormat = "#N";
        }
        channel ethernetline extends ThruputMeteringChannel
        {
            delay = 0.1us;
            datarate = 10Mbps;
            thruputDisplayFormat = "#N";
        }
    submodules:
        FTP_client: StandardHost {
            @display("p=552,62");
        }
        configurator: IPv4NetworkConfigurator {
            @display("p=316,42");
        }
        FTP_server: StandardHost {
            @display("p=89,62");
        }
        VOIP_client: StandardHost {
            @display("p=89,255");
        }
        VOIP_client2: StandardHost {
            @display("p=546,255");
        }
        router1: Router {
            @display("p=226,163");
        }
        router2: Router {
            @display("p=421,163");
        }
```

Pro konfiguraci síťové adresace slouží prvek IPv4NetworkConfigurator, pokud necháme jeho nastavení na původních parametrech, přiřadí se IP adresy v celé síti automaticky.

K dokončení návrhu už zbývá pouze propojit prvky sítě. Jedná se o plně duplexní spojení s parametry kanálů z předchozího bodu.

```
    connections:
        router2.ethg++ <--> ethernetline <--> FTP_client.ethg++;
        router2.ethg++ <--> ethernetline <--> VOIP_client2.ethg++;
        router1.ethg++ <--> ethernetline <--> VOIP_client.ethg++;
        router1.ethg++ <--> ethernetline <--> FTP_server.ethg++;
        router2.pppg++ <--> PPP <--> router1.pppg++;}
```

Tímto je vámi vytvořená síť kompletně zprovozněna k provádění simulací, nyní tedy musíme určit parametry a průběh simulace. Jelikož soubor NED je statický, je vhodné v něm definovat pouze základní parametry, které nebudeme dále měnit ani nijak upravovat. Pro nastavení průběhu simulace a síťového provozu slouží konfigurační soubor s příponou .ini.

Do složky s projektem vytvoříme nový soubor, **New » Initialization File (ini) »** název můžeme ponechat, **omnetpp.ini » Next » Empty Ini File » Finish**.

Doplníme název sítě, ke které chceme konfigurační soubor přiřadit (**network = DiffservNetwork**). Pod něj vložíme následující kód:

```
debug-on-errors = true
tkenv-plugin-path = ../inet/etc/plugins
sim-time-limit = 300s
```

Tím povolíme výpis problému při chybně nastavené simulaci, nastavíme cestu k potřebným doplňkům a nastavíme maximální trvání simulace na 5 minut.

Pro potřeby simulace je možné v jednom souboru vytvářet několik různých konfigurací, k vybrání požadované budeme vyzváni po spuštění simulace. Pro vytvoření nové konfigurace se používá syntaxe [Config nazev_konfigurace]. Hlavní částí je sekce [General], která obsahuje nastavení společná pro všechny vytvořené konfigurace, jejím nastavením začneme.

Nadefinujeme FTP klienta a FTP server:

```
**.FTP_server.numTcpApps = 1
**.FTP_server.tcpApp[0].typename = "TCPBasicClientApp"
**.FTP_server.tcpApp[0].connectAddress = "FTP_client"
**.FTP_server.tcpApp[0].connectPort = 20
**.FTP_server.tcpApp[0].thinkTime = 0.001s
**.FTP_server.tcpApp[0].idleInterval = 0.001s
**.FTP_server.tcpApp[0].startTime = 0s
**.FTP_server.tcpApp[0].requestLength = 5MiB
**.FTP_server.tcpApp[0].numRequestsPerSession = 1
**.FTP_server.tcpApp[0].reconnectInterval = 1s

**.FTP_client.numTcpApps = 1
**.FTP_client.tcpApp[0].typename = "TCPGenericSrvApp"
**.FTP_client.tcpApp[0].localPort = 20
```

Nadefinujeme provedení VOIP hovoru s dobou trvání 100 sekund:

```
**.VOIP_client*.numUdpApps = 1
**.VOIP_client.udpApp[*].typename = "SimpleVoIPSender"
**.VOIP_client.udpApp[*].destAddress = "VOIP_client2"
**.VOIP_client.udpApp[0].destPort = 2000
#**.VOIP_client.udpApp[1].destPort = 2001
**.VOIP_client.udpApp[*].startTime = 0s
**.VOIP_client.udpApp[*].stopTime = 100s
**.VOIP_client.udpApp[*].talkPacketSize = 500B
**.VOIP_client.udpApp[*].packetizationInterval = 5ms

**.VOIP_client2.udpApp[*].typename = "SimpleVoIPReceiver"
**.VOIP_client2.udpApp[0].localPort = 2000
#**.VOIP_client2.udpApp[1].localPort = 2001
```

Dále vytvoříme novou konfiguraci - **[Config FIFO]**, ve které nastavíme typ fronty FIFO pro směrovače a kapacitu 80 paketů. Při překročení této hranice jsou všechny nově příchozí pakety zahazovány.

```
[Config FIFO]
**.router*.ppp[0].queueType = "DropTailQueue"
**.router*.ppp[0].queue.frameCapacity = 80
```

Vypracování 2. část - PQ

Vytvoříme další konfiguraci **[Config PQ]**, ve které nastavíme mechanizmus zacházení s pakety pomocí prioritních front. Vložíme do ní následující kód, ve kterém je potřeba upravit cestu k odkazovaným souborům (ingressTCType a queueType) podle názvu projektu:

```
[Config PQ]
description = Priority queueing
**.router1.*.ingressTCType = "nazev_projektu.TrafficConditioner"
**.ingressTC.classifier.filters = xmldoc("filters.xml")
**.router1.*.queueType = "nazev_projektu.PQQueue"
```

Tím jsme na vstupní rozhraní routeru 1 nastavili mechanizmus pro značkování provozu TrafficConditioner, tento soubor nyní vytvoříme a nastavíme jeho parametry. **New » Network Description File (NED)**, soubor pojmenujeme **TrafficConditioner.ned » Next » Empty NED File » Finish**. Do nově vytvořeného vložíme následující kód, který umožní rozdělení datového toku do tříd, podle kterých přidělí příchozím paketům odpovídající označení:

```
import inet.linklayer.contract.ITrafficConditioner;
import inet.networklayer.diffserv.DSCPMarker;
import inet.networklayer.diffserv.MultiFieldClassifier;

module TrafficConditioner like ITrafficConditioner
{   parameters:
        @display("i=block/classifier");
    gates:
        input in;
        output out;
    submodules:
        classifier: MultiFieldClassifier {
            @display("p=61,110");         }
        marker: DSCPMarker {
            dscps = "EF AF11 AF21 AF31 AF41 BE";
            @display("p=192,110");        }
    connections:
        in --> classifier.in;
        for i=1..5 {classifier.outs++ --> marker.in++;}
        classifier.defaultOut --> marker.in++;
        marker.out --> out;
}
```

Nyní vytvoříme soubor, ve kterém definujeme systém prioritní fronty, **New » Network Description File (NED)**, soubor pojmenujeme **PQQueue.ned » Next » Empty NED File » Finish.** Prioritní fronta je zapojena na výstupu směrovače, pakety se řadí do jednotlivých front podle přiřazené značky a na výstupní port se posílá přednostně fronta s nejvyšší prioritou (EF). Mechanizmus je popsán následujícím kódem, který vložíme do souboru PQQueue.ned.

```
import inet.common.queue.DropTailQueue;
import inet.linklayer.contract.ITrafficConditioner;
import inet.common.queue.PriorityScheduler;
import inet.networklayer.diffserv.BehaviorAggregateClassifier;
import inet.networklayer.diffserv.DSCPMarker;
import inet.networklayer.diffserv.MultiFieldClassifier;
import inet.common.queue.IOutputQueue;

module PQQueue like IOutputQueue
{    parameters:
        int frameCapacity = default(20);
        @display("bgb=661,722");
    gates:  input in;
            output out;
    types:
    submodules:
        efQueue: DropTailQueue {
            frameCapacity = 50;
            @display("p=365,91");         }
        af1xQueue: DropTailQueue {
            frameCapacity = 10;
            @display("p=365,200");         }
        af2xQueue: DropTailQueue {
            frameCapacity = frameCapacity;
            @display("p=365,305");         }
        af3xQueue: DropTailQueue {
            frameCapacity = frameCapacity;
            @display("p=365,397");         }
        af4xQueue: DropTailQueue {
            frameCapacity = frameCapacity;
            @display("p=365,513");         }
        beQueue: DropTailQueue {
            frameCapacity = frameCapacity;
            @display("p=365,604");         }
        priority: PriorityScheduler {
            @display("p=574,337");         }
        dscpMarker: DSCPMarker {
            dscps = "EF";
            @display("p=216,123");         }
        dscpMarker1: DSCPMarker {
            dscps = "AF11";
            @display("p=216,211");         }
        dscpMarker2: DSCPMarker {
            dscps = "AF21";
            @display("p=216,287");         }
        dscpMarker3: DSCPMarker {
            dscps = "AF31";
            @display("p=217,361");         }
        dscpMarker4: DSCPMarker {
            dscps = "AF41";
            @display("p=217,439");         }
        dscpMarker5: DSCPMarker {
            dscps = "BE";
            @display("p=216,525");         }
        Classifier: BehaviorAggregateClassifier{
            dscps = "EF AF11 AF21 AF31 AF41";
            @display("p=55,326");         }
```

Nakonec ještě vložíme kód popisující propojení vložených prvků:

```
    connections:
        in --> Classifier.in;
        Classifier.outs++ --> dscpMarker.in++;
        Classifier.outs++ --> dscpMarker1.in++;
        Classifier.outs++ --> dscpMarker2.in++;
        Classifier.outs++ --> dscpMarker3.in++;
        Classifier.outs++ --> dscpMarker4.in++;
        Classifier.defaultOut --> dscpMarker5.in++;
        dscpMarker.out --> efQueue.in;
        dscpMarker1.out --> af1xQueue.in;
        dscpMarker2.out --> af2xQueue.in;
        dscpMarker3.out --> af3xQueue.in;
        dscpMarker4.out --> af4xQueue.in;
        dscpMarker5.out --> beQueue.in;
        efQueue.out --> priority.in++;
        af1xQueue.out --> priority.in++;
        af2xQueue.out --> priority.in++;
        af3xQueue.out --> priority.in++;
        af4xQueue.out --> priority.in++;
        beQueue.out --> priority.in++;
        priority.out --> out;
}
```

Vypracování 3. část - WFQ

Soubor uložíme a přepneme se zpět do konfiguračního souboru omnetpp.ini. Vytvoříme další konfiguraci **[Config WFQ]** a vložíme kód pro zavedení mechanizmu WFQ na router1:

```
[Config WFQ]
description = Weighted fair queueing
**.router1.*.ingressTCType = "nazev_ projektu.TrafficConditioner"
**.ingressTC.classifier.filters = xmldoc("filters.xml")
**.router1.*.queueType = "nazev_projektu.WFQueue"
**.router1.**.wrr.weights = "12 2 0 0 0 1"
```

Tím jsme nastavili značkování příchozích paketů na routeru 1, podle přidělené značky jsou posílány do odpovídající fronty, pro které se parametrem „weights“ nastavuje jejich váhový koeficient, který funguje na principu přidělování tokenů jednotlivým třídám v počtu odpovídající vahám a jeden token odpovídá jednomu odeslanému paketu. První číslo je pro frontu s nejvyšší prioritou (EF), kterou přiřadíme pro přenos hovoru VOIP. Druhé číslo je pro frontu s přenosem souboru FTP a poslední číslo je pro základní třídu (BE), do které patří neoznačkovaný datový provoz.

Vytvoříme nový soubor **WFQueue.ned**, který reprezentujte systém váhových front na výstupním portu směrovače:

```
import inet.common.queue.DropTailQueue;
import inet.common.queue.IOutputQueue;
import inet.common.queue.PriorityScheduler;
import inet.common.queue.WRRScheduler;
import inet.networklayer.diffserv.BehaviorAggregateClassifier;

module WFQueue like IOutputQueue
{   parameters:
        int frameCapacity = default(20);
    gates:  input in;
             output out;
    submodules:
        classifier: BehaviorAggregateClassifier {
            dscps = "EF AF11 AF21 AF31 AF41";
            @display("p=41,284");         }
        efQueue: DropTailQueue {
            frameCapacity = 100;
            @display("p=195,95");         }
        af1xQueue: DropTailQueue {
            frameCapacity = 10;
            @display("p=195,224");         }
        af2xQueue: DropTailQueue {
            frameCapacity = frameCapacity;
            @display("p=195,329");         }
        af3xQueue: DropTailQueue {
            frameCapacity = frameCapacity;
            @display("p=195,421");         }
        af4xQueue: DropTailQueue {
            frameCapacity = frameCapacity;
            @display("p=195,537");         }
        beQueue: DropTailQueue {
            frameCapacity = frameCapacity;
            @display("p=195,628");         }
        wrr: WRRScheduler {
            @display("p=384,368");         }
    connections:
        in --> classifier.in;
        classifier.outs++ --> efQueue.in;
        classifier.outs++ --> af1xQueue.in;
        classifier.outs++ --> af2xQueue.in;
        classifier.outs++ --> af3xQueue.in;
        classifier.outs++ --> af4xQueue.in;
        classifier.defaultOut --> beQueue.in;
        efQueue.out --> wrr.in++;
        af1xQueue.out --> wrr.in++;
        af2xQueue.out --> wrr.in++;
        af3xQueue.out --> wrr.in++;
        af4xQueue.out --> wrr.in++;
        beQueue.out --> wrr.in++;
        wrr.out --> out;
}
```

Nyní už zbývá jen rozdělení síťového provozu do požadovaných tříd, vytvoříme nový soubor **New » File »** s názvem **filters.xml** a vložíme do něj následující podmínky:

```
<filters>
  <filter destPort="2000" gate="0"/>
  <filter destPort="20" gate="1"/>
</filters>
```

Rozdělení zapsané v souboru filters.xml je použito pro oba mechanizmy řízení odesílání, forma zápisu je jednoduchá, brána 0 označuje data s nejvyšší prioritou. Přiřazení datového toku probíhá podle cílového portu (port 2000 využívá VOIP hovor), jako přiřazovací parametr lze použít i zdrojový port, zdrojovou nebo cílovou adresu, protokol a podobně.

3. Simulace projektu

Před prvním spuštěním simulace ještě musíme sestavit projekt, kliknutím pravým tlačítkem myši na vaši složku **» Build Configurations » Build Selected… »** zaškrtneme **gcc-debug » OK**.

Po dokončení operace je projekt připraven k simulaci, kliknutím pravým tlačítkem myši na soubor **omnetpp.ini » Run As » OMNeT++ Simulation »** z kontextové nabídky vybereme požadovanou konfiguraci.

Stisknutím ikony **Run** (nebo klávesou **F5**) se simulace spustí, v s simulačním okně probíhá výpis poslaných zpráv, které se zobrazují také v animaci sítě. U každého spoje je po spuštění simulace číselná hodnota, která vypisuje počet přenesených paketů. Pro rychlejší dokončení lze stisknout ikonu **Express (F7)**. Pro změnu výběru simulovaného scénáře není zapotřebí otevírat nové okno simulace, stačí stisknout nabídku **File » Set Up a Configuration** a vybrat požadovaný scénář.

Po dokončení simulace se zobrazí výpis trvání v sekundách, tato doba odpovídá době potřebné k dokončení přenosu souboru pomocí FTP, hodnotu porovnáme pro všechny scénáře.

Po dokončení všech simulací se přesuneme do složky **results**, kde se uložili soubory s výsledky. Dvojklikem otevřeme libovolný soubor pro každý scénář a potvrdíme vytvoření souboru s příponou **.anf**, nabízené jméno ponecháme. V souborech s příponou .anf můžeme prohlížet výsledky provedených simulací, v záložce **Browse Data** máme na výběr z výsledků zobrazených vektorově v grafu a skalárními hodnotami.

Zobrazíme si statistiku zpoždění VOIP aplikace. **Vectors »** v poli **statistic name filter** vybereme **endToEndDelay:vector**. Zjistíme průměrnou hodnotu zpoždění (sloupec **Mean**) a jitter (**StdDev**). Kliknutím pravým tlačítkem myši » **Plot**, můžeme zobrazit průběh zpoždění v grafické podobě. Porovnáme tyto parametry pro všechny tři scénáře.

Dále si zobrazíme statistiku obousměrného zpoždění FTP paketů. **Vectors »** v poli **statistic name filter** vybereme **smoothed RTT**. Opět porovnáme pro všechny tři scénáře.

Zobrazíme počet zahozených paketů. **Scalars »** v poli **statistic name filter** vybereme **dropPk:count**, opět porovnáme. U výsledků PQ a WFQ můžeme vidět, počet zahozených paketů jednotlivých front a tím i typ třídy síťového provozu.

Zkuste měnit nastavení váhových koeficientů WFQ a pozorujte, jak se změní parametry přenosu, například nastavením vah na 1 1 1 1 1 1 se nastaví systém FQ se spravedlivou obsluhou všech front.

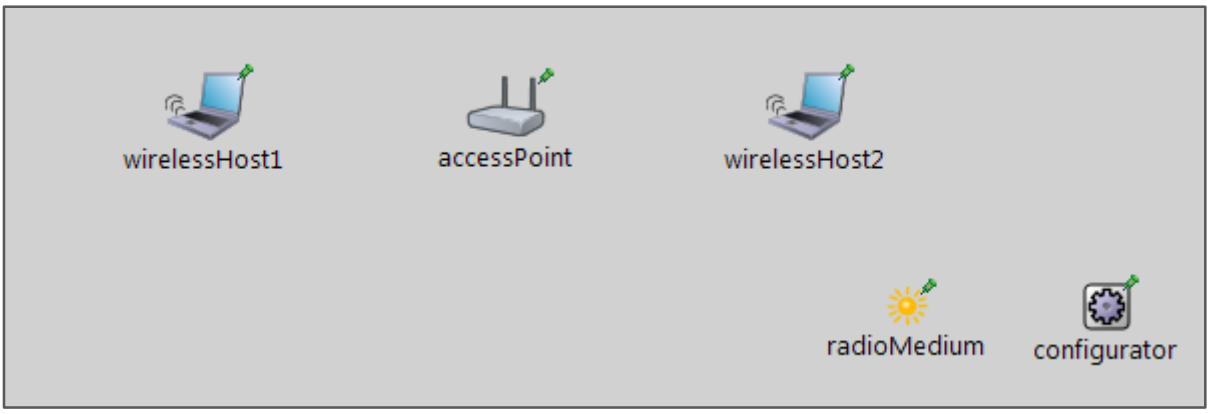
Zapněte generování druhého VOIP hovoru (nemusíte dopisovat kód, stačí smazat znaky komentáře „#“ v souboru omnetpp.ini), vytvořte pro něj značkování (v souboru filters.xml pro cílový port 2001) a sledujte vliv na kvalitu přenosu.

Otázky:

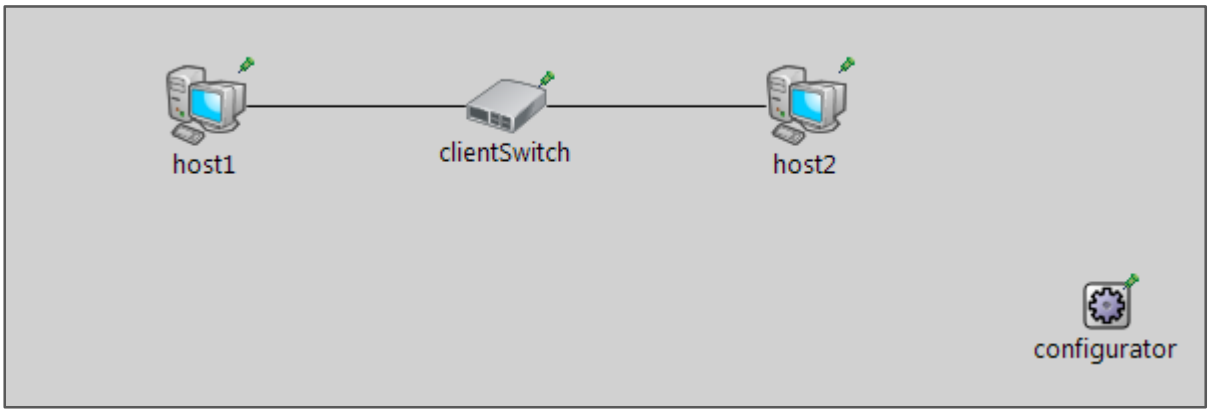
1. Jaký způsob řízení odesílání paketů umožňuje fronta FIFO?
2. Jaký vliv mají váhová čísla na řízení odesílání paketů mechanizmem WFQ?
3. Proč docházelo u mechanizmů PQ a WFQ k zahazování paketů? A proč se zahazovaly pouze pakety FTP přenosu?

C. Laboratorní úloha č.3 – WLAN vs. Ethernet

Cílem laboratorní úlohy je srovnání v současnosti nejvyužívanějších technologií k realizaci počítačových sítí, kabelové spojení technologií ethernet dle standardu IEEE 802.3 a spojení bezdrátové, využívající k přenosu radiový signál, dle standardu IEEE 802.11. Úloha srovnává parametry přenosu dat v obou technologiích a zaměřuje se také na přenos řídících dat potřebných k navázání a udržení spojení. Dále se blíže sledují možnosti bezdrátové sítě, například podpora pro řízení kvality služeb a vliv vzdálenosti na chybovost přenášených dat. K simulaci je využito jednoduché zapojení dvou navzájem komunikujících klientů pomocí přístupového bodu, respektive přepínače. Topologii bezdrátové sítě lze vidět na obrázku 1. Zapojení sítě ethernet lze vidět na obrázku 2.



Obrázek 1: Topologie bezdrátové sítě



Obrázek 2: Topologie kabelové sítě

1. Teoretický úvod

Síť Ethernet byla vyvinuta v roce 1973 spolčeností Xerox, přenosová rychlost byla 2,94 Mbit/s a spojení probíhalo pouze pomocí koaxiálního kabelu v poloduplexním módu. Ethernet procházel dalším vývojem a byl uveden pod standardem IEEE 802.3, který byl během let mnohokrát modifikován a v současnosti existuje celá řada standardů rozvíjející původní návrh. Bylo dosaženo podstatných vylepšení, zejména přenosové rychlosti, která dosahuje až 100 Gbit/s a obousměrné komunikace v plně duplexním módu. Provedení kabelu může být pomocí optovláken a přenos je uskutečněn pomocí světelného záření. Nebo metalickými kabely, nejpoužívanější je nestíněná kroucená dvoulinka s označením UTP, kde jsou jednotlivé bity reprezentovány elektrickými signály. Jako konektor se standardně používá osmi-pinový typ RJ-45, který poskytuje dvojici vodičů pro vysílání v obou směrech. Standard IEEE 802.3 kromě fyzické vrstvy popisuje také vrstvu spojovou, která je druhou vrstvou referenčního modelu ISO/OSI. Datová jednotka na této vrstvě se nazývá rámec a má pevně definovanou strukturu. Minimální velikost rámce je 64B a maximální velikost 1518B a musí obsahovat pole s cílovou a zdrojovou fyzickou adresa. Část obsahující přenášená data může mít velikost v rozsahu od 46B-1500B, při menší velikost dat musí být použita výplň k doplnění do 46B, aby byla dosaženo minimální velikosti rámce.

Vývoj standardu IEEE 802.11 definující bezdrátové sítě započal v roce 1997, maximální přenosová rychlost se rovnala 2 Mbit/s, v této podobě se dnes již nepoužívá. Opět docházelo k dalšímu vývoji a bylo vydáno několik dalších standardů, které výrazně vylepšují přenosovou rychlost (až 866,7 Mbit/s se standardem 802.11ac) a zavádí nové technologie, například MIMO, tato technologie umožňuje současný přenos několika vysílacích i přijímacích antén na různých kanálech a tím je možné ještě několikanásobně zvýšit přenosovou rychlost. Bezdrátové sítě nabízí oproti sítím kabelovým výhodu ve snadné tvorbě a modifikaci topologie sítě, lze přidávat nebo odpojovat stanice ze sítě bez fyzického zásahu a je umožněno také připojení mobilních klientů. Bezdrátové sítě musí nutně obsahovat implementaci metod řízení k přístupovému médiu, tou je standardně metoda CSMA/CA, která snižuje riziko kolizí při současném vysílání více stanic. Radiový signál je vysílán volně šířen prostorem a zprávy jsou tak přijímány všemi stanicemi v dosahu vysílače a stanice, pro které nejsou data určena, je zahazují. Tím dochází ke snížení teoretické propustnosti přenosového kanálu.

2. Vypracování

Založení nového projektu

Spustíme program **omnetpp.exe**. Pokud se objeví nabídka pro vybrání pracovního adresáře, tento formulář potvrdíme beze změny (vybraná cesta musí být: omnetpp-4.6\samples), jinak vytvořený projekt nebude fungovat.

K vytvoření nového projektu klikneme na **File » New » OMNeT++ Project…** Projekt pojmenujeme VUT loginem, dále stiskneme **Next » Empty project » Finish**. V okně Project Explorer se vytvořila námi pojmenovaná složka, ve které budou uloženy všechny části projektu.

Nejprve je zapotřebí přiřadit k projektu odkazy na zdrojové soubory, se kterými se bude dále pracovat. To provedeme kliknutím pravým tlačítkem na vytvořenou složku » **Properties » Project References »** vybereme položku **inet » OK**.

Pro vytvoření projektu jsou zapotřebí dva základní soubory. Network Description File, s příponou .ned, který definuje topologii sítě, použité technologie a síťové prvky. Druhým je Initialization File, přípona .ini, který určuje průběh a parametry simulace.

Klikněte pravým tlačítkem myši na vaši složku a vyberte **New » Network Description File (NED)**, pojmenujte ji **Network.ned » Next » Empty NED File » Finish**.

V souboru NED lze pracovat ve dvou režimech, v grafickém nebo kódovém. Mezi nimi se lze přepínat v levém dolním rohu volbou Design/Source. Jsou navzájem propojené, pokud tedy přidáme prvky sítě z palety, vytvoří se k nim patřičný kód a naopak vložením kódu lze vytvářet celou síť.

Vypracování 1. část – Zahájení komunikace a řídící data WLAN sítě

Nejprve musíme importovat do projektu prvky, se kterými se bude pracovat. Do souboru Network.ned vložíme následující kód:

```
import inet.networklayer.configurator.ipv4.IPv4NetworkConfigurator;
import inet.node.ethernet.EtherSwitch;
import inet.node.inet.StandardHost;
import inet.physicallayer.contract.packetlevel.IRadioMedium;
import ned.DatarateChannel;
import inet.node.inet.WirelessHost;
import inet.node.wireless.AccessPoint;
```

Vytvoříme síť pojmenovanou WirelessNetwork, která bude obsahovat dva klienty, komunikující pomocí bezdrátového přístupového bodu.

```
network WirelessNetwork
{
    parameters:
        @display("bgb=600,200");
        string radioMediumType;

    submodules:
        configurator: IPv4NetworkConfigurator {
            @display("p=550,150");
        }
        radioMedium: <radioMediumType> like IRadioMedium {
            @display("p=450,150");
        }
        wirelessHost2: WirelessHost {
            @display("p=400,50");
        }
        accessPoint: AccessPoint {
            @display("p=250,50");
        }
        wirelessHost1: WirelessHost {
            @display("p=100,50");
        }
}
```

Pro konfiguraci síťové adresace slouží prvek IPv4NetworkConfigurator, pokud necháme jeho nastavení na původních parametrech, přiřadí se IP adresy v celé síti automaticky. Prvek radioMedium slouží k definování parametrů bezdrátového spojení, které budeme nastavovat až v následujících krocích.

Tímto je návrh sítě kompletně dokončen, ale k provádění simulací ještě musíme určit parametry a průběh simulace. Jelikož soubor NED je statický, je vhodné v něm definovat pouze základní parametry, které nebudeme dále měnit ani nijak upravovat. Pro nastavení průběhu simulace a síťového provozu slouží konfigurační soubor s příponou .ini, který nyní musíme vytvořit.

Do složky s projektem vložíme nový soubor, **New » Initialization File (ini) »** název můžeme ponechat, **omnetpp.ini » Next » Empty Ini File » Finish**.

Přepneme se do textového módu (tlačítko Source v levém dolním rohu). Smažeme nabídku (**network =**). A místo toho vložíme následující kód:

```
debug-on-errors = true
tkenv-plugin-path = ../inet/etc/plugins
sim-time-limit = 20s
```

Tím povolíme výpis chybových hlášení při nesprávně nastavené simulaci, nastavíme cestu k potřebným doplňkům a nastavíme maximální trvání simulace na 20 sekund.

Pro potřeby simulace je možné v jednom souboru vytvářet několik různých konfigurací, k vybrání požadované budeme vyzvání po spuštění simulace. Pro vytvoření nové konfigurace se používá syntaxe [Config nazev_konfigurace]. Hlavní částí je sekce [General], která obsahuje nastavení společná pro všechny vytvořené konfigurace, jejím nastavením začneme.

Nejprve musíme nastavit parametry bezdrátové sítě a nadefinovat přenosové médium.

```
# Nastavení WLAN
**.wlan[*].opMode = "g"
**.wlan[*].bitrate = 11Mbps
**.accessPoint.wlan[*].mac.address = "09:00:0F:00:00:0A"
**.wirelessHost1.wlan[*].mac.address = "11:C0:00:00:00:11"
**.wirelessHost2.wlan[*].mac.address = "22:00:0D:B1:00:22"
**.wlan[*].mac.maxQueueSize = 50
**.wlan[*].mac.rtsThresholdBytes = 2346B
**.wlan[*].radio.transmitter.power = 4mW
**.wlan[*].radio.transmitter.headerBitLength = 100b
**.wlan[*].radio.carrierFrequency = 2.4GHz
**.wlan[*].radio.bandwidth = 2MHz
**.wlan[*].radio.receiver.sensitivity = -85dBm
**.wlan[*].radio.receiver.snirThreshold = 4dB
**.wlan[*].radio.receiver.energyDetection = -85dBm
**.accessPoint.wlan[*].mgmt.beaconInterval = 100ms
**.accessPoint.wlan[*].mgmt.numAuthSteps = 4
**.wlan[*].radio.receiver.errorModelType = "APSKErrorModel"
**.wlan[*].mgmt.ssid = "WLAN1"
**.wirelessHost*.wlan[*].agent.default_ssid = ""

# Nastavení přenosového média
*.radioMediumType = "RadioMedium"
*.radioMedium.backgroundNoiseType = "IsotropicScalarBackgroundNoise"
*.radioMedium.pathLossType = "FreeSpacePathLoss"
*.radioMedium.analogModelType = "ScalarAnalogModel"
*.radioMedium.propagationType = "ConstantTimePropagation"
*.radioMedium.backgroundNoise.power = -100dBm
```

Tímto kódem nastavíme bezdrátové vysílání standardem IEEE 802.11g na frekvenci 2,4GHz a s přenosovou rychlostí 11Mbit/s. Také jsme určili parametry vysílače a přijímače, vysílací výkon, citlivost a další. A nakonec definujeme parametry přenosu, který bude prováděn radiovým signálem.

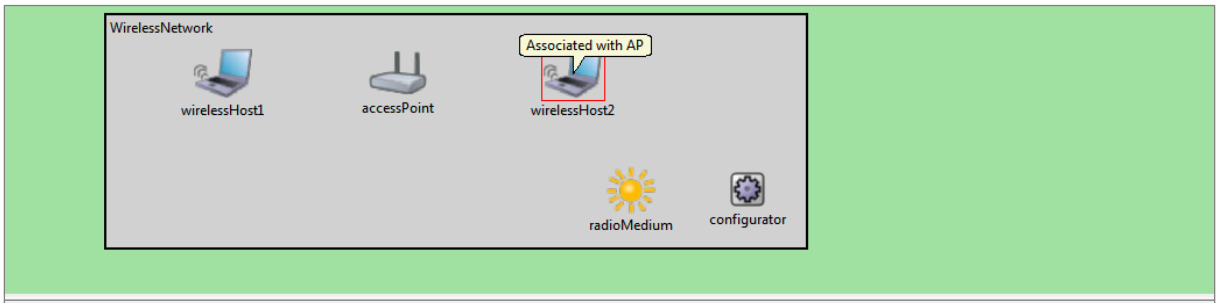
Vytvoříme novou konfiguraci s názvem Wireless_network, kterou vložíme na konec konfiguračního souboru omnetpp.ini a pouze k ní přiřadíme námi vytvořenou síť.

```
[Config Wireless_network]
network = WirelessNetwork
```

Před spuštěním simulace je zapotřebí sestavit projekt, to provedeme kliknutím pravým tlačítkem myši na složku s projektem **» Build Configurations » Build Selected… »** zaškrtneme **gcc-debug » OK**.

Po dokončení operace je projekt připraven k simulaci, kliknutím pravým tlačítkem myši na konfigurační soubor **omnetpp.ini » Run As** » **OMNeT++ Simulation »** z kontextové nabídky můžeme vybrat požadovanou konfiguraci, v tomto případě **Wireless_network**.

Stisknutím ikony **Run** (nebo klávesou **F5**) se simulace spustí, v s simulačním okně probíhá výpis poslaných zpráv, které se zobrazují také v animaci sítě. Můžeme vidět, jaké zprávy se posílají při navazování bezdrátového spojení, a také řídící rámce pravidelně rozesílané přístupovým bodem.



Event#	Time	Src/Dest	Name	Info	
#66	0.238438170822	accessPoint --> wirelessHost2	Beacon	WLAN beacon	duration=0.046ms
#66	0.238438170822	accessPoint --> wirelessHost1	Beacon	WLAN beacon	duration=0.046ms
#83	0.287069108919	wirelessHost2 --> accessPoint	ProbeReq	WLAN probe request	duration=0.042ms
#83	0.287069108919	wirelessHost2 --> wirelessHost1	ProbeReq	WLAN probe request	duration=0.042ms
#98	0.287641108919	accessPoint --> wirelessHost2	ProbeResp	WLAN probe response	duration=0.249ms
#98	0.287641108919	accessPoint --> wirelessHost1	ProbeResp	WLAN probe response	duration=0.249ms
#110	0.287900108919	wirelessHost2 --> accessPoint	ACK	WLAN ack 10-00-00-00-00-00	duration=0.304ms
#110	0.287900108919	wirelessHost2 --> wirelessHost1	ACK	WLAN ack 10-00-00-00-00-00	duration=0.304ms
#127	0.293768234527	wirelessHost1 --> wirelessHost2	ProbeReq	WLAN probe request	duration=0.042ms
#127	0.293768234527	wirelessHost1 --> accessPoint	ProbeReq	WLAN probe request	duration=0.042ms
#142	0.294120234527	accessPoint --> wirelessHost2	ProbeResp	WLAN probe response	duration=0.249ms
#142	0.294120234527	accessPoint --> wirelessHost1	ProbeResp	WLAN probe response	duration=0.249ms
#154	0.294379234527	wirelessHost1 --> wirelessHost2	ACK	WLAN ack 10-00-00-00-00-00	duration=0.304ms
#154	0.294379234527	wirelessHost1 --> accessPoint	ACK	WLAN ack 10-00-00-00-00-00	duration=0.304ms
#177	0.338438170822	accessPoint --> wirelessHost2	Beacon	WLAN beacon	duration=0.046ms
#177	0.338438170822	accessPoint --> wirelessHost1	Beacon	WLAN beacon	duration=0.046ms
#195	0.438438170822	accessPoint --> wirelessHost2	Beacon	WLAN beacon	duration=0.046ms
#195	0.438438170822	accessPoint --> wirelessHost1	Beacon	WLAN beacon	duration=0.046ms
#213	0.538438170822	accessPoint --> wirelessHost2	Beacon	WLAN beacon	duration=0.046ms
#213	0.538438170822	accessPoint --> wirelessHost1	Beacon	WLAN beacon	duration=0.046ms
#234	0.587069108919	wirelessHost2 --> accessPoint	Auth	WLAN auth	duration=0.217ms
#234	0.587069108919	wirelessHost2 --> wirelessHost1	Auth	WLAN auth	duration=0.217ms
#247	0.587296108919	accessPoint --> wirelessHost2	ACK	WLAN ack 0A-AA-00-00-00-05	duration=0.304ms
#247	0.587296108919	accessPoint --> wirelessHost1	ACK	WLAN ack 0A-AA-00-00-00-05	duration=0.304ms
#262	0.588010108919	accessPoint --> wirelessHost2	Auth	WLAN auth	duration=0.217ms
#262	0.588010108919	accessPoint --> wirelessHost1	Auth	WLAN auth	duration=0.217ms
#275	0.588237108919	wirelessHost2 --> accessPoint	ACK	WLAN ack 10-00-00-00-00-00	duration=0.304ms
#275	0.588237108919	wirelessHost2 --> wirelessHost1	ACK	WLAN ack 10-00-00-00-00-00	duration=0.304ms
#290	0.588651108919	wirelessHost2 --> accessPoint	Auth	WLAN auth	duration=0.217ms
#290	0.588651108919	wirelessHost2 --> wirelessHost1	Auth	WLAN auth	duration=0.217ms
#303	0.588878108919	accessPoint --> wirelessHost2	ACK	WLAN ack 0A-AA-00-00-00-05	duration=0.304ms
#303	0.588878108919	accessPoint --> wirelessHost1	ACK	WLAN ack 0A-AA-00-00-00-05	duration=0.304ms
#318	0.589572108919	accessPoint --> wirelessHost2	Auth-OK	WLAN auth	duration=0.217ms
#318	0.589572108919	accessPoint --> wirelessHost1	Auth-OK	WLAN auth	duration=0.217ms
#335	0.589799108919	wirelessHost2 --> accessPoint	ACK	WLAN ack 10-00-00-00-00-00	duration=0.304ms
#335	0.589799108919	wirelessHost2 --> wirelessHost1	ACK	WLAN ack 10-00-00-00-00-00	duration=0.304ms
#350	0.590533108919	wirelessHost2 --> accessPoint	Assoc	WLAN assoc req	duration=0.224ms
#350	0.590533108919	wirelessHost2 --> wirelessHost1	Assoc	WLAN assoc req	duration=0.224ms
#363	0.590767108919	accessPoint --> wirelessHost2	ACK	WLAN ack 0A-AA-00-00-00-05	duration=0.304ms
#363	0.590767108919	accessPoint --> wirelessHost1	ACK	WLAN ack 0A-AA-00-00-00-05	duration=0.304ms
#378	0.591501108919	accessPoint --> wirelessHost2	AssocResp-OK	WLAN assoc resp	duration=0.232ms
#378	0.591501108919	accessPoint --> wirelessHost1	AssocResp-OK	WLAN assoc resp	duration=0.232ms

Obrázek 3: Navázání komunikace s přístupovým bodem

Vypracování 2. část – Ethernet vs. WLAN, porovnání parametrů

Nyní přidáme do naší bezdrátové sítě datový provoz, následující kód vložíme do sekce **[Config Wireless_network]**:

```
**.numUdpApps = 1
**.wirelessHost2.udpApp[0].typename = "UDPVideoStreamSvr"
**.wirelessHost2.udpApp[0].videoSize = 10MiB
**.wirelessHost2.udpApp[0].localPort = 1000
**.wirelessHost2.udpApp[0].sendInterval = 1ms
**.wirelessHost2.udpApp[0].packetLen = 1200B

**.wirelessHost1.udpApp[0].typename = "UDPVideoStreamCli"
**.wirelessHost1.udpApp[0].serverAddress = "wirelessHost2"
**.wirelessHost1.udpApp[0].localPort = 2000
**.wirelessHost1.udpApp[0].serverPort = 1000
**.wirelessHost1.udpApp[0].startTime = 1
```

Tímto nastavíme streamování videa o velikosti 10MB mezi klienty, stejně tak jej nastavíme i pro síť s ethernetem, kterou nyní vytvoříme.

Novou síť pojmenujeme **WiredNetwork**, bude mít stejnou topologií jako síť bezdrátová z předchozí části, nyní však budou klienti připojeni technologií ethernet, s přenosovou rychlostí 10Mbit/s. Na konec souboru **Network.ned** přidáme následující kód:

```
network WiredNetwork
{   parameters:
        @display("bgb=600,200");

    types:
        channel LAN10 extends DatarateChannel
        {   delay = 0.1us;
            datarate = 10Mbps;
            ber = 1E-8;
        }

    submodules:
        configurator: IPv4NetworkConfigurator {
            @display("p=550,150");
        }
        host1: StandardHost {
            @display("p=100,50");
        }
        host2: StandardHost {
            @display("p=400,50");
        }
        switch: EtherSwitch {
            @display("p=250,50");
        }
    connections:
        host1.ethg++ <--> LAN10 <--> clientSwitch.ethg++;
        clientSwitch.ethg++ <--> LAN10 <--> host2.ethg++;
}
```

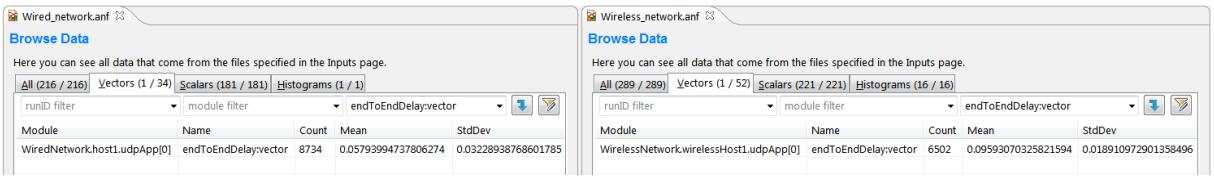
V souboru **omnetpp.ini** vytvoříme novou konfiguraci s názvem **Wired_network**, vložíme do ní odkaz na nově vytvořenou síť a nadefinujeme datový provoz se stejnými parametry jako v síti bezdrátové:

```
[Config Wired_network]
network = WiredNetwork

**.host*.numUdpApps = 1
**.host2.udpApp[0].typename = "UDPVideoStreamSvr"
**.host2.udpApp[0].videoSize = 10MiB
**.host2.udpApp[0].localPort = 1000
**.host2.udpApp[0].sendInterval = 1ms
**.host2.udpApp[0].packetLen = 1200B

**.host1.udpApp[0].typename = "UDPVideoStreamCli"
**.host1.udpApp[0].serverAddress = "host2"
**.host1.udpApp[0].localPort = 2000
**.host1.udpApp[0].serverPort = 1000
**.host1.udpApp[0].startTime = 1s
```

Nyní můžeme provést simulaci, spustíme postupně oba nově vytvořené scénáře (**Wireless_network, Wired_network**). Urychlení simulace lze provést ikonou **FAST** (klávesa **F6**) nebo **EXPRESS (F7)**. Pro změnu výběru simulovaného scénáře není zapotřebí otevírat nové okno simulace, stačí stisknout nabídku **File » Set Up a Configuration** a vybrat požadovaný scénář.

Po úspěšném dokončení simulací obou scénářů můžeme okno simulátoru zavřít a přesunout se k souborům s výsledky simulací. Ty jsou uloženy ve složce **results**, umístěné v našem projektu a pojmenované dle názvu scénáře. Dvojklikem otevřeme libovolný soubor pro každý scénář a potvrdíme vytvoření souboru s příponou **.anf**, nabízené jméno ponecháme. V souborech s příponou .anf můžeme prohlížet výsledky provedených simulací, v záložce **Browse Data** máme na výběr z výsledků zobrazených vektorově v grafu a skalárními hodnotami.

K porovnání vybereme například statistiku zpoždění přenosu streamovaného videa. Vybereme záložku **Vectors »** v poli **statistic name filter** vybereme **endToEndDelay:vector**. Zde můžeme vidět průměrnou hodnotu zpoždění (sloupec Mean) a jitter (sloupec StdDev). Porovnáme tyto hodnoty mezi oběma scénáři.



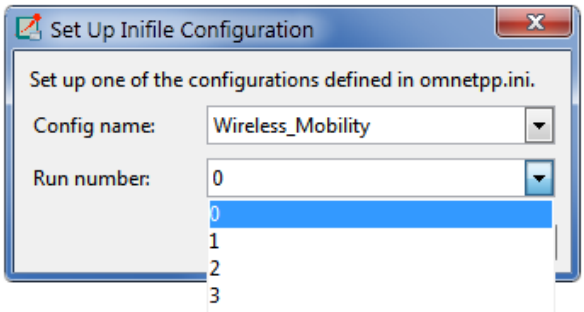
Obrázek 4: Okno s výslednými hodnotami parametru zpoždění

Vypracování 3. část – Parametry přenosu v závislosti na vzdálenosti

V konfiguračním souboru **omnetpp.ini** vytvoříme další scénář s názvem [Config Wireless_Mobility]. Do kterého nastavíme rostoucí vzdálenost klienta (wirelessHost2) od přístupového bodu s krokem po padesáti metrech, od 150m do 300m. A budeme sledovat vliv vzdálenosti na chybovost přenášených dat.

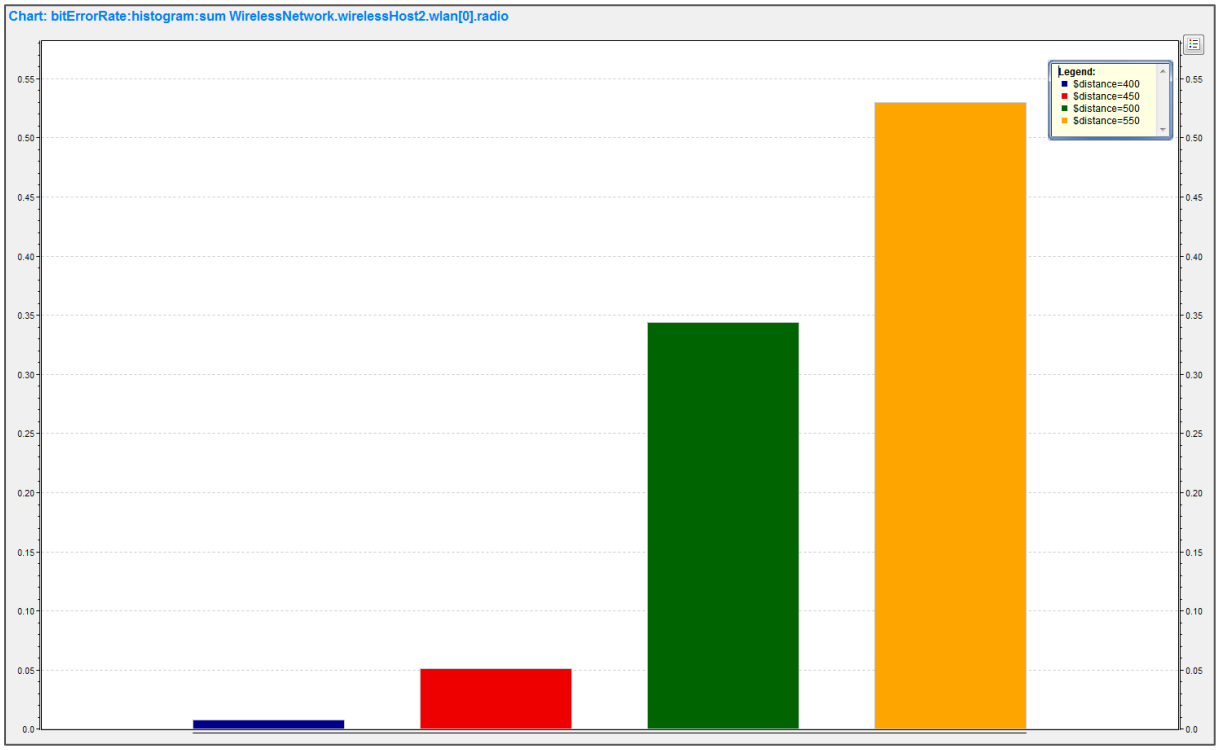
```
[Config Wireless_Mobility]
extends = Wireless_network
**.constraintAreaMinX = 0m
**.constraintAreaMinY = 0m
**.constraintAreaMinZ = 0m
**.constraintAreaMaxX = 800m
**.constraintAreaMaxY = 600m
**.constraintAreaMaxZ = 0m

**.mobilityType = "StationaryMobility"
**.mobility.initFromDisplayString = false
**.wirelessHost1.mobility.initialX = 100m
**.wirelessHost1.mobility.initialY = 50m
**.wirelessHost2.mobility.initialX = ${distance=400..550 step 50}m
**.wirelessHost2.mobility.initialY = 50m
**.accessPoint.mobility.initialX = 250m
**.accessPoint.mobility.initialY = 50m
**.mobility.initialZ = 0m
```

Před spuštěním simulace budeme nově vyzvání také k výběru pořadového čísla simulace, viz obrázek 3. Každé z nich reprezentuje jinou hodnotu nastavovaného parametru, tedy v tomto případě měnící se vzdálenost mezi klientem a přístupovým bodem. Vybereme tedy scénář **Wireless_mobility** a postupně provedeme simulaci pro všechna pořadová čísla (0 až 3). K urychlení průběhu simulace můžeme využít ikonu **Express (F7).**



Obrázek 5: Výběr pořadového čísla simulace

Po dokončení simulací se přesuneme do složky **results**, kde se vytvořili nové soubory s výsledky pojmenované **Wireless_Mobility**, dvojklikem na jeden z nich otevřeme okno s výsledky. Přepneme se na záložku **Browse Data » Scalars**. V poli **module filter** zadáme **WirelessNetwork.wirelessHost2.wlan[0].radio,** v poli **statistic name filter** vybereme **bitErrorRate:histogram:sum**. Zobrazí se nám hodnoty bitové chybovosti přenášených dat a porovnáme závislost na vzdálenosti. Všechny čtyři položky označíme a kliknutím pravým tlačítkem myši **» Plot** zobrazíme grafické znázornění chybovosti. Legendu pro popis položek lze zobrazit ikonou v pravém horním rohu grafu.



Obrázek 6: Bitová chybovost v závislosti na vzdálenosti bezdrátové komunikace

Dále si zobrazíme počet úspěšně doručených paketů pro jednotlivé vzdálenosti. Přepneme se zpět na záložku **Browse Data » Scalars**. Do pole **module filter** zadáme **WirelessNetwork.wirelessHost1.wlan[0].mac** a v poli **statistic name filter** vybereme **passedUpPk:count**, zobrazí se nám hodnoty úspěšně doručených paketů, které můžeme opět zobrazit do grafu jako v předchozím případě. Ve scénáři s největší vzdáleností od přístupového se již nepodařilo ani navázat spojení k bezdrátové komunikaci a můžeme vidět, že počet přenesených paketů je výrazně nižší a jedná se pouze o řídící data bezdrátové sítě (beacon rámce a pokusy o navázání spojení).

Nyní vytvoříme nový scénář v souboru **omnetpp.ini** s názvem [Config Wired_Mobility], ve kterém zjistíme vliv vzdálenosti na parametry přenosu dat kabelovým spojením a porovnáme s výsledky v síti bezdrátové. Nastavíme dvě různé vzdálenosti od přepínače, 150 a 300m, druhá jmenovaná již neumožnila kvůli vysoké chybovosti navázání spojení v bezdrátové síti.

```
[Config Wired_Mobility]
extends = Wired_network
**.constraintAreaMinX = 0m
**.constraintAreaMinY = 0m
**.constraintAreaMinZ = 0m
**.constraintAreaMaxX = 800m
**.constraintAreaMaxY = 600m
**.constraintAreaMaxZ = 0m

**.mobilityType = "StationaryMobility"
**.mobility.initFromDisplayString = false
**.host1.mobility.initialX = 100m
**.host1.mobility.initialY = 50m
**.host2.mobility.initialX = ${distance=400..550 step 150}m
**.host2.mobility.initialY = 50m
**.mobility.initialZ = 0m
```

Spustíme simulaci a vybereme scénář Wired_mobility, který provedeme pro dvě různé vzdálenosti (Run number 0 a 1). Opět můžeme simulaci urychlit ikonou **Express (F7).** Po dokončení otevřeme soubor s výsledky a zobrazíme si stejnou statistiku přenesených paketů, jako v předchozím bodě a hodnoty porovnáme. Záložka **Browse Data » Scalars**. Do pole **module filter** zadáme **WiredNetwork.host1.udp** a v poli **statistic name filter** vybereme **passedUpPk:count**, můžeme vidět, že hodnoty se s rostoucí vzdáleností prakticky nemění.

Vypracování 4. část – Srovnání parametrů při vysoké zátěži sítě

V této části zvýšíme datový tok v síti a porovnáme vliv na zpoždění přenosu v obou sítích. Vložením následujícího kódu na konec konfiguračního souboru **omnetpp.ini** vytvoříme dvě shodné konfigurace pro obě naše sítě, do kterých tímto přidáme druhý videostream s obdobnými parametry jako ten původní.

```
[Config Wired_network_High_load]
extends = Wired_network

**.host1.numUdpApps = 2
**.host1.udpApp[1].typename = "UDPBasicApp"
**.host1.udpApp[1].destAddresses = "host2"
**.host1.udpApp[1].packetName = "VideoStream_2"
**.host1.udpApp[1].destPort = 4000
**.host1.udpApp[1].startTime = 1.01s
**.host1.udpApp[1].stopTime = 10.0s
**.host1.udpApp[1].messageLength = 1200B
**.host1.udpApp[1].sendInterval = 0.001s
**.host2.numUdpApps = 2
**.host2.udpApp[1].typename = "UDPSink"
**.host2.udpApp[1].localPort = 4000

[Config Wireless_network_High_load]
extends = Wireless_network

**.wirelessHost1.numUdpApps = 2
**.wirelessHost1.udpApp[1].typename = "UDPBasicApp"
**.wirelessHost1.udpApp[1].destAddresses = "wirelessHost2"
**.wirelessHost1.udpApp[1].packetName = "VideoStream_2"
**.wirelessHost1.udpApp[1].destPort = 4000
**.wirelessHost1.udpApp[1].startTime = 1.01s
**.wirelessHost1.udpApp[1].stopTime = 10.0s
**.wirelessHost1.udpApp[1].messageLength = 1200B
**.wirelessHost1.udpApp[1].sendInterval = 0.001s
**.wirelessHost2.numUdpApps = 2
**.wirelessHost2.udpApp[1].typename = "UDPSink"
**.wirelessHost2.udpApp[1].localPort = 4000
```

Spustíme simulaci pro oba nově vytvořené scénáře (Wireless_network_High_load a Wired_network_High_load), opět ji můžeme urychlit ikonou **Express (F7)**. Po dokončení otevřeme soubory s výsledky a přepneme se na záložku **Browse Data » Vectors »** v poli **statistic name filter** vybereme **endToEndDelay:vector**. Porovnáme hodnoty pro obě sítě, ve sloupci Mean je zobrazena průměrná hodnota zpoždění v sekundách, můžeme vidět, že v bezdrátové síti došlo vlivem vysokého zatížení k výraznému zhoršení zpoždění oproti síti propojené ethernetem.

Vypracování 5. část – EDCA

Nyní vyzkoušíme použití mechanizmu pro řízení kvality služeb v bezdrátových sítích EDCA, která umožňuje rozdělení datového provozu do tříd podle čísla portu a zajistit kvalitativní zvýhodnění služeb ve třídách s vyšší prioritou. Pro každou třídu jsou definovány různé parametry při soutěžení o přístupu k přenosovému kanálu, pravděpodobnost přístupu je nejvyšší pro třídy vysokou prioritou.

V konfiguračním souboru **omnetpp.ini** vytvoříme novou konfiguraci s názvem [Config Wireless_network_EDCA], která bude obsahovat následující kód:

```
[Config Wireless_network_EDCA]
extends = Wireless_network_High_load

**.wlan[*].mac.EDCA = true
**.wlan[*].classifierType = "ExampleQoSClassifier"
**.wlan[*].mac.cwMinData = 15
**.wlan[*].mac.cwMaxData = 1023
```

V síti jsou dva shodné videostreamy z předchozí části, které plně vytěžují přenosový kanál bezdrátové sítě a oba dosahují velkého zpoždění. Zavedením mechanizmu EDCA zvýhodníme jeden stream posílající video na portu 5000 a budeme sledovat, jak se změní výsledné hodnoty zpoždění.

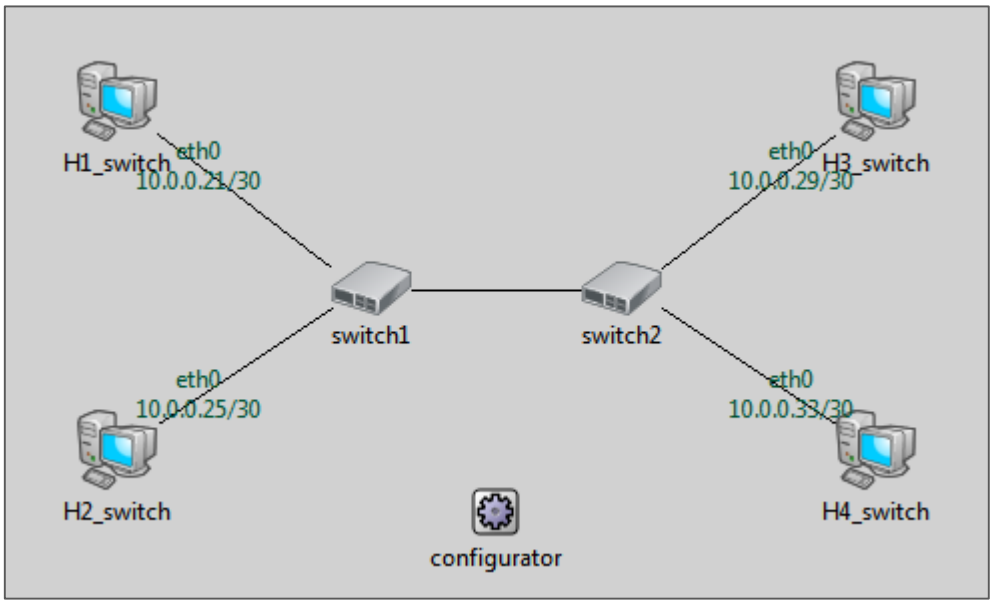
Provedeme simulaci scénáře **Wireless_network_EDCA**. Otevřeme soubor s výsledky a přepneme se na záložku **Browse Data » Vectors »** v poli **statistic name filter** vybereme **endToEndDelay:vector**. Ve sloupci **Mean** je zobrazena průměrná hodnota zpoždění obou videostreamů. Můžeme porovnat tyto hodnoty s těmi předchozími ze scénáře bez řízení kvality (Wireless_network_High_load).

Otázky:

1. Jak jsou pojmenovány řídící rámce WLAN sítí?
2. Na základě čeho lze v bezdrátových sítích rozlišit data do tříd v systému EDCA?

D. Laboratorní úloha č.4 – Aktivní síťové prvky

Cílem laboratorní úlohy je srovnání nejpoužívanějších aktivních síťových prvků, které slouží k vytváření počítačových sítí a umožňují komunikaci mezi koncovými stanicemi. V úloze jsou k propojení sítě použity rozbočovače, přepínače a směrovače, každý z těchto prvků pracuje na jiné vrstvě referenčního modelu ISO/OSI. Tím jsou dány hlavní rozdíly v jejich funkci při zpracování a přeposílání příchozích dat, které lze sledovat ve vytvořené simulaci. Ke srovnání síťových prvků je sledován parametr zpoždění přenosu souborů službou FTP a množství rozesílaných dat na jednotlivých linkách. Pro simulaci poslouží jednoduchá síť se čtyřmi klienty. Topologii sítě propojené přepínači lze vidět na obrázku 1, ostatní sítě jsou obdobné, pouze je k propojení koncových stanic využito rozbočovačů, respektive směrovačů.



Obrázek 1: Topologie sítě propojené přepínači

1. Teoretický úvod

Způsob funkce a principy, které jsou aplikovány při zpracování a přenosu zpráv v rámci počítačových sítí vychází z rozdělení jednotlivých úkonů do skupin definovaných v referenčním modelu ISO/OSI, případně ve zjednodušeném modelu TCP/IP. Složení obou modelů lze vidět v tabulce 1. Pro úspěšné provedení a zpracování přenosu informace, dochází k postupnému zpracování datové jednotky danou vrstvou a předání její sousední vrstvě. Pro samotný přenos dat jsou nejdůležitější tři nejnižší vrstvy, fyzická, spojová a síťová. Zbylé čtyři vrstvy, respektive aplikační vrstva modelu TCP/IP mají za úkol především práci s datovým obsahem zpráv a jsou obvykle záležitostí uživatelských aplikací.

Na třech nejnižších vrstvách operují aktivní síťové prvky, které zprostředkovávají přenos informace skrze komunikační síť. Podle typu zařízení mohou vykonávat i mnoho dalších funkcí, jako řízení přístupu ke sdílenému médiu, zajištění kvality služeb nebo směrování datového toků sítí s nalezením nejlepší cesty. Množina operací, které jsou schopny dané prvky provádět, vychází právě z principu rozdělení komunikačního modelu na jednotlivé vrstvy. Základním uzlem síťové vrstvy je směrovač, který poskytuje nejrozsáhlejší možnosti při zpracování přijatých nebo odesílaných zpráv, jelikož musí také zvládat operace obou nižších vrstev. Na druhé vrstvě operuje přepínač, který nabízí menší možnosti nastavení a řízení provozu, ale výhodou jsou nižší výpočetní nároky. Rozbočovač pracuje pouze na nejnižší fyzické vrstvě a jejich možnosti jsou tedy velmi omezené, výhodou je však jednoduchost nastavení, minimální výpočetní nároky a minimální zpoždění přenášených dat.

Tabulka 1: Referenční model ISO/OSI a TCP/IP

<table>
<tr><td>7. Aplikační vrstva</td><td rowspan="5">4. Aplikační vrstva</td></tr>
<tr><td>6. Prezentační vrstva</td></tr>
<tr><td>5. Relační vrstva</td></tr>
<tr><td>4. Transportní vrstva</td></tr>
<tr><td>3. Síťová vrstva</td><td>3. Síťová vrstva</td></tr>
<tr><td>2. Spojová vrstva</td><td>2. Spojová vrstva</td></tr>
<tr><td>1. Fyzická vrstva</td><td>1. Fyzická vrstva</td></tr>
</table>

2. Vypracování

Založení nového projektu

Spustíme program **omnetpp.exe**. Pokud se objeví nabídka pro vybrání pracovního adresáře, tento formulář potvrdíme beze změny (vybraná cesta musí být: omnetpp-4.6\samples), jinak vytvořený projekt nebude fungovat.

K vytvoření nového projektu klikneme na **File » New » OMNeT++ Project…** Projekt pojmenujeme VUT loginem, dále stiskneme **Next » Empty project » Finish**. V okně Project Explorer se vytvořila námi pojmenovaná složka, ve které budou uloženy všechny části projektu. Nejprve je zapotřebí přiřadit k projektu odkazy na zdrojové soubory, se kterými se bude dále pracovat. To provedeme kliknutím pravým tlačítkem na vytvořenou složku » **Properties » Project References »** vybereme položku **inet » OK**.

Pro vytvoření projektu jsou zapotřebí dva základní soubory. Network Description File, s příponou .ned, který definuje topologii sítě, použité technologie a síťové prvky. Druhým je Initialization File, přípona .ini, který určuje průběh a parametry simulace.

Klikněte pravým tlačítkem myši na vaši složku a vyberte **New » Network Description File (NED)**, pojmenujte ji **Network.ned » Next » Empty NED File » Finish**.

V souboru NED lze pracovat ve dvou režimech, v grafickém nebo kódovém. Mezi nimi se lze přepínat v levém dolním rohu volbou Design/Source. Jsou navzájem propojené, pokud tedy přidáme prvky sítě z palety, vytvoří se k nim patřičný kód a naopak vložením kódu lze vytvářet celou síť.

Vypracování 1. část – Aktivní síťové prvky

Nejprve musíme importovat do projektu prvky, se kterými se bude pracovat. Do souboru Network.ned vložíme následující kód:

```
import inet.linklayer.ethernet.EtherHub;
import inet.networklayer.configurator.ipv4.IPv4NetworkConfigurator;
import inet.node.ethernet.EtherSwitch;
import inet.node.inet.Router;
import inet.node.inet.StandardHost;
import inet.common.misc.ThruputMeteringChannel;
```

Nejprve vytvoříme síť pojmenovanou Hub, která bude obsahovat čtyři klienty, propojené dvěma rozbočovači, nastavíme přenosovou linku typu ethernet s přenosovou rychlostí 10Mbit/s pro její realizaci zvolíme model ThruputMeteringChannel, který umožňuje zobrazení počtu přenesených paketů jednotlivých linek přímo v simulaci.

```
network Hub
{    parameters:
        @display("bgb=466,313");
    types:
        channel ethernet extends ThruputMeteringChannel
        {   delay = 0.1us;
            datarate = 10Mbps;
            thruputDisplayFormat = "N";
        }
    submodules:
        hub1: EtherHub {
            @display("p=167,148");
            gates:  ethg[3];
        }
        hub2: EtherHub {
            @display("p=292,148");
            gates:  ethg[3];
        }
        H1_hub: StandardHost {
            @display("p=40,56");
            gates:  ethg[1];
        }
        H2_hub: StandardHost {
            @display("p=40,230");
            gates:  ethg[1];
        }
        H3_hub: StandardHost {
            @display("p=419,56");
            gates:  ethg[1];
        }
        H4_hub: StandardHost {
            @display("p=419,230");
            gates:  ethg[1];
        }
        configurator: IPv4NetworkConfigurator {
            @display("p=228,276");
        }
    connections:
        H1_hub.ethg[0] <--> ethernet <--> hub1.ethg[0];
        H2_hub.ethg[0] <--> ethernet <--> hub1.ethg[1];
        H3_hub.ethg[0] <--> ethernet <--> hub2.ethg[0];
        H4_hub.ethg[0] <--> ethernet <--> hub2.ethg[1];
        hub1.ethg[2] <--> ethernet <--> hub2.ethg[2];
}
```

Pro konfiguraci síťové adresace slouží prvek IPv4NetworkConfigurator, jeho nastavení ponecháme na původních parametrech, čímž se IP adresy přiřadí pro všechny prvky v síti automaticky.

Nyní vytvoříme síť, v níž k propojení klientů použijeme přepínače, ta bude mít stejnou topologii i parametry, pouze nahradíme huby za switche.

```
network Switch
{   parameters:
        @display("bgb=466,313");
    types:
        channel ethernet extends ThruputMeteringChannel
        {   delay = 0.1us;
            datarate = 10Mbps;
            thruputDisplayFormat = "N";
        }
    submodules:
        switch1: EtherSwitch {
            @display("p=168,155");
            gates:  ethg[3];
        }
        switch2: EtherSwitch {
            @display("p=293,155");
            gates:  ethg[3];
        }
        H1_switch: StandardHost {
            @display("p=41,62");
            gates:  ethg[1];
        }
        H2_switch: StandardHost {
            @display("p=41,236");
            gates:  ethg[1];
        }
        H3_switch: StandardHost {
            @display("p=420,62");
            gates:  ethg[1];
        }
        H4_switch: StandardHost {
            @display("p=420,236");
            gates:  ethg[1];
        }
        configurator: IPv4NetworkConfigurator {
            @display("p=230,267");
        }
    connections:
        H1_switch.ethg[0] <--> ethernet <--> switch1.ethg[0];
        H2_switch.ethg[0] <--> ethernet <--> switch1.ethg[1];
        H3_switch.ethg[0] <--> ethernet <--> switch2.ethg[0];
        H4_switch.ethg[0] <--> ethernet <--> switch2.ethg[1];
        switch1.ethg[2] <--> ethernet <--> switch2.ethg[2];
}
```

A nakonec vytvoříme síť se směrovači:

```
network Router
{   parameters:
        @display("bgb=466,313");
    types:
        channel ethernet extends ThruputMeteringChannel
        {   delay = 0.1us;
            datarate = 10Mbps;
            thruputDisplayFormat = "N";
        }
    submodules:
        router1: Router {
            @display("p=168,156");
            gates:  ethg[3];
        }
        router2: Router {
            @display("p=293,156");
            gates:  ethg[3];
        }
        H1_router: StandardHost {
            @display("p=44,62");
            gates:  ethg[1];
        }
        H2_router: StandardHost {
            @display("p=44,236");
            gates:  ethg[1];
        }
        H3_router: StandardHost {
            @display("p=420,62");
            gates:  ethg[1];
        }
        H4_router: StandardHost {
            @display("p=420,236");
            gates:  ethg[1];
        }
        configurator: IPv4NetworkConfigurator {
            @display("p=228,267");
        }
    connections:
        H1_router.ethg[0] <--> ethernet <--> router1.ethg[0];
        H2_router.ethg[0] <--> ethernet <--> router1.ethg[1];
        H3_router.ethg[0] <--> ethernet <--> router2.ethg[0];
        H4_router.ethg[0] <--> ethernet <--> router2.ethg[1];
        router1.ethg[2] <--> ethernet <--> router2.ethg[2];
}
```

Tímto je návrh našich sítí zkompletován, ale k provádění simulací ještě musíme určit parametry a průběh simulace. Jelikož soubor NED je statický, je vhodné v něm definovat pouze základní parametry, které nebudeme dále měnit ani nijak upravovat. Pro nastavení průběhu simulace a síťového provozu slouží konfigurační soubor s příponou .ini, který nyní musíme vytvořit.

Do složky s projektem vložíme nový soubor, **New » Initialization File (ini) »** název můžeme ponechat, **omnetpp.ini » Next » Empty Ini File » Finish**.

Přepneme se do textového módu (tlačítko Source v levém dolním rohu). Smažeme nabídku (**network** =). A místo toho vložíme následující kód:

```
debug-on-errors = true
tkenv-plugin-path = ../inet/etc/plugins
sim-time-limit = 20s
**.macType = "EtherMAC"
**.mss = 1460
```

Tím povolíme výpis chybových hlášení při nesprávně nastavené simulaci, nastavíme cestu k potřebným doplňkům a nastavíme maximální trvání simulace na 20 sekund. Také nastavíme maximální možnou délku TCP segmentu, 1460B.

Nyní nadefinujeme datový provoz, kterým bude přenos souborů, pomocí protokolu FTP, mezi stanicemi H1 a H4

```
**.H1*.numTcpApps = 2
**.H4*.numTcpApps = 2
**.H1*.tcpApp[0].typename = "TCPBasicClientApp"
**.H1*.tcpApp[0].connectPort = 20
**.H1*.tcpApp[0].thinkTime = 0.001s
**.H1*.tcpApp[0].idleInterval = 0.001s
**.H1*.tcpApp[0].startTime = 0s
**.H1*.tcpApp[0].numRequestsPerSession = 5
**.H1*.tcpApp[0].requestLength = 10MiB
**.H1*.tcpApp[0].reconnectInterval = 1s
**.H4*.tcpApp[0].typename = "TCPBasicClientApp"
**.H4*.tcpApp[0].connectPort = 20
**.H4*.tcpApp[0].thinkTime = 0.001s
**.H4*.tcpApp[0].idleInterval = 0.001s
**.H4*.tcpApp[0].startTime = 0s
**.H4*.tcpApp[0].numRequestsPerSession = 5
**.H4*.tcpApp[0].requestLength = 10MiB
**.H4*.tcpApp[0].reconnectInterval = 1s
**.H4*.tcpApp[1].typename = "TCPGenericSrvApp"
**.H4*.tcpApp[1].localPort = 20
**.H1*.tcpApp[1].typename = "TCPGenericSrvApp"
**.H1*.tcpApp[1].localPort = 20
```

Pro potřeby simulace je možné v jednom souboru vytvářet několik různých konfigurací, k vybrání požadované budeme vyzvání po spuštění simulace. Hlavní částí je sekce [General], která obsahuje nastavení společná pro všechny vytvořené konfigurace. Pro vytvoření nové konfigurace se používá syntaxe [Config nazev_konfigurace].

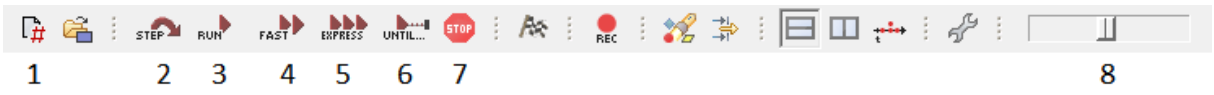
Vytvoříme tři nové konfigurace, a každé z nich přiřadíme jednu z vytvořených sítí, pojmenujeme je dle aktivního síťového prvku, který je v dané síti použit (Hub, Switch, Router):

```
[Config Hub]
network = Hub

**.H1_hub.tcpApp[0].connectAddress = "H4_hub"
**.H4_hub.tcpApp[0].connectAddress = "H1_hub"

[Config Switch]
network = Switch

**.H1_switch.tcpApp[0].connectAddress = "H4_switch"
**.H4_switch.tcpApp[0].connectAddress = "H1_switch"

[Config Router]
network = Router

**.H1_router.tcpApp[0].connectAddress = "H4_router"
**.H4_router.tcpApp[0].connectAddress = "H1_router"
```

Nyní můžeme přejít k simulaci, ale nejprve je zapotřebí sestavit projekt, to provedeme kliknutím pravým tlačítkem myši na složku s projektem » **Build Configurations** » **Build Selected…** » zaškrtneme **gcc-debug** » **OK**.

Po dokončení operace je projekt připraven k simulaci, kliknutím pravým tlačítkem myši na konfigurační soubor **omnetpp.ini** » **Run As** » **OMNeT++ Simulation** » z kontextové nabídky můžeme vybrat požadovanou konfiguraci, postupně spustíme všechny tři.

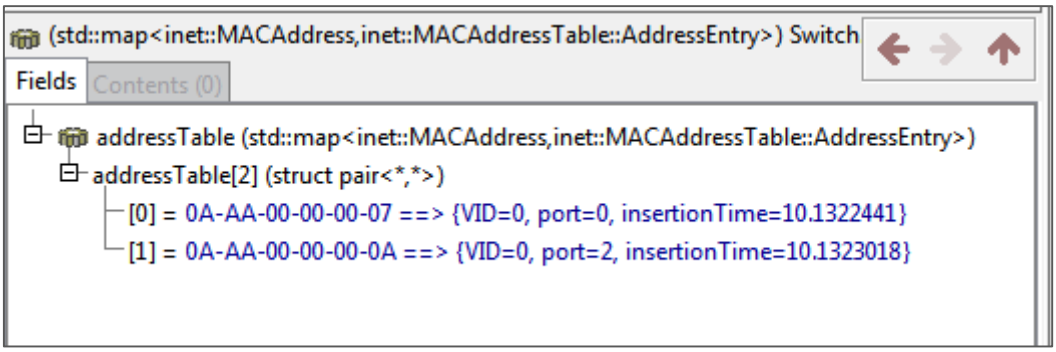
Průběh simulace lze ovládat pomocí panelu s ikonami viz Obrázek 2. Ikony 2 až 5 spustí simulaci s různou rychlostí, ikonou Until (6) se lze posunout na požadovaný čas v simulaci a ikonou Stop (7) se simulace pozastaví. Rychlost zobrazování simulace lze také regulovat posuvníkem Animation speed (8). Pro výběr jiného scénáře můžeme použít ikonu 1.



Obrázek 2: Ovládací panel simulačního okna

Spustíme simulaci stisknutím ikony Run (3) a posuvníkem (8) nastavíme takovou rychlost, abychom mohli pozorovat posílání zpráv v grafické části simulace, můžeme také vidět výpis zpráv v dolní části obrazovky. Budeme sledovat, jakým způsobem probíhá posílání zpráv sítí v každém ze tří námi vytvořených sítí. Po přenosu dat se u každého spoje v animaci zobrazuje počet přenesených paketů jednotlivými rozhraními, u nevyužitích linek zůstává IP adresa a rozhraní. Můžeme sledovat, na kterých linkách a jaké množství paketů se rozesílá v jednotlivých sítích, připomeňme si, že přenos dat v naší síti probíhá pouze mezi koncovými stanicemi H1 a H4.

Po kliknutí na síťový prvek můžeme v levém dolním rohu vidět parametry a bližší informace, například směrovací tabulky, které si zobrazíme pro jednotlivé prvky a porovnáme jejich obsah. Pro zobrazení směrovací tabulky routeru klikneme myší na jeden ze směrovačů a v okně s obsahem (obvykle v levém dolním rohu) vybereme **routingTable » routes**. Pro zobrazení směrovací tabulky switche vybereme **macTable » addressTable**. Pokud klikneme na rozbočovač, můžeme vidět, že ten neuchovává žádné informace.



Obrázek 3: Obsah směrovací tabulky přepínače

Vypracování 2. část – Zpoždění přenosu

Pro snadnější zobrazení výsledků vytvoříme novou síť, která bude obsahovat všechny tři předchozí, následující kód přidáme do souboru **Network.ned.**

```
network Network
{    parameters:          @display("bgb=1428,313");
     types: channel ethernet extends ThruputMeteringChannel
          {    delay = 0.1us;
               datarate = 10Mbps;
               thruputDisplayFormat = "N";
          }
     submodules:
          switch1: EtherSwitch {@display("p=650,154");    gates: ethg[3];}
          switch2: EtherSwitch {@display("p=775,154");    gates: ethg[3];}
          hub1: EtherHub {         @display("p=167,148");    gates: ethg[3];}
          hub2: EtherHub {       @display("p=292,148");    gates: ethg[3];}
          router1: Router {      @display("p=1134,155");   gates: ethg[3];}
          router2: Router {       @display("p=1259,155");  gates: ethg[3];}
          H1_switch: StandardHost {@display("p=523,61");  gates: ethg[1];}
          H2_switch: StandardHost {@display("p=523,235");gates: ethg[1];}
          H3_switch: StandardHost {@display("p=902,61");  gates: ethg[1];}
          H4_switch: StandardHost {@display("p=902,235");gates: ethg[1];}
          H1_hub: StandardHost {       @display("p=40,56");gates: ethg[1];}
          H2_hub: StandardHost {    @display("p=40,230");  gates: ethg[1];}
          H3_hub: StandardHost {    @display("p=419,56");  gates: ethg[1];}
          H4_hub: StandardHost {    @display("p=419,230");gates: ethg[1];}
          H1_router: StandardHost {@display("p=1010,61");gates: ethg[1];}
          H2_router: StandardHost {@display("p=1010,235");gates:ethg[1];}
          H3_router: StandardHost {@display("p=1386,61");  gates:ethg[1];}
          H4_router: StandardHost {@display("p=1386,235");gates:ethg[1];}
          configurator: IPv4NetworkConfigurator {@display("p=712,266"); }

     connections:
          H1_switch.ethg[0] <--> ethernet <--> switch1.ethg[0];
          H2_switch.ethg[0] <--> ethernet <--> switch1.ethg[1];
          H3_switch.ethg[0] <--> ethernet <--> switch2.ethg[0];
          H4_switch.ethg[0] <--> ethernet <--> switch2.ethg[1];
          H1_hub.ethg[0] <--> ethernet <--> hub1.ethg[0];
          H2_hub.ethg[0] <--> ethernet <--> hub1.ethg[1];
          H3_hub.ethg[0] <--> ethernet <--> hub2.ethg[0];
          H4_hub.ethg[0] <--> ethernet <--> hub2.ethg[1];
          H1_router.ethg[0] <--> ethernet <--> router1.ethg[0];
          H2_router.ethg[0] <--> ethernet <--> router1.ethg[1];
          H3_router.ethg[0] <--> ethernet <--> router2.ethg[0];
          H4_router.ethg[0] <--> ethernet <--> router2.ethg[1];
          hub1.ethg[2] <--> ethernet <--> hub2.ethg[2];
          switch1.ethg[2] <--> ethernet <--> switch2.ethg[2];
          router1.ethg[2] <--> ethernet <--> router2.ethg[2];
      }
```

V souboru **omnetpp.ini** vytvoříme novou konfiguraci s názvem Network, vložením kódu na konec souboru:

```
[Config Network]
network = Network

**.H1_hub.tcpApp[0].connectAddress = "H4_hub"
**.H4_hub.tcpApp[0].connectAddress = "H1_hub"
**.H1_switch.tcpApp[0].connectAddress = "H4_switch"
**.H4_switch.tcpApp[0].connectAddress = "H1_switch"
**.H1_router.tcpApp[0].connectAddress = "H4_router"
**.H4_router.tcpApp[0].connectAddress = "H1_router"
```

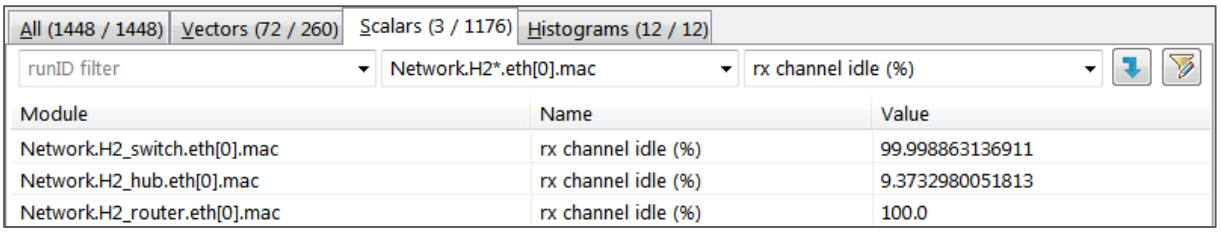
Nyní můžeme provést simulaci, vybereme nově vytvořenou konfiguraci (**Network**) a provedeme simulaci až do jejího ukončení. Pro urychlení můžeme použít ikonu **Express** (5).

Po úspěšném dokončení simulace můžeme okno simulátoru zavřít a přesunout se k souborům s výsledky simulací. Ty jsou uloženy ve složce **results**, umístěné v našem projektu a pojmenované dle názvu scénáře. Dvojklikem na jeden ze souborů s názvem Network výsledky vytvoříme a potvrdíme vytvoření souboru s příponou **.anf**, nabízené jméno ponecháme. V souborech s příponou .anf můžeme prohlížet výsledky provedených simulací, v záložce **Browse Data** máme na výběr z výsledků zobrazených vektorově v grafu a skalárními hodnotami.

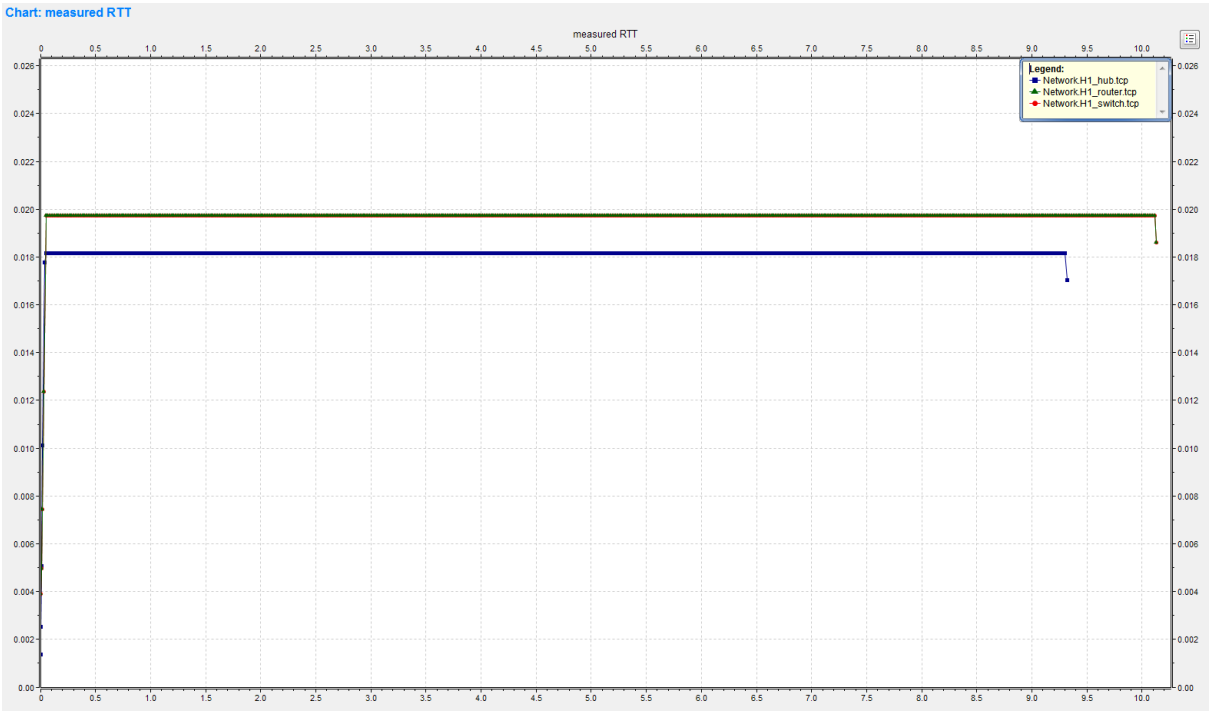
K porovnání vybereme statistiku zpoždění přenosu FTP dat. Vybereme záložku **Vectors** » do pole **module filter** zadáme **Network.H1*.tcp** a v poli **statistic name filter** vybereme **measured RTT**. Nyní můžeme porovnat průměrnou hodnotu zpoždění (sloupec Mean) jednotlivých síťových prvků.

Výsledky můžeme zobrazit také do grafu, označíme všechny tři položky a klikneme pravým tlačítkem myši » **Plot**. V pravém horním rohu grafu si můžeme zobrazit legendu. Který síťový prvek dokončil přenos souborů v nejkratším čase?

Dále si zobrazíme procento nečinnosti linky ke klientské stanici H2, která nepřijímá ani neodesílá žádná data. Přepneme se na záložku **Scalars** » do pole **module filter** zadáme **Network.H2*.eth[0].mac** a v poli **statistic name filter** vybereme **rx channel idle (%)**. Tím zobrazíme procentuální hodnotu nečinnosti dané linky, můžeme vidět, že pro hub je procento nečinnosti výrazně nižší.



Obrázek 4: Zobrazení nečinnosti přenosové linky



Obrázek 5: Grafické zobrazení parametru zpoždění

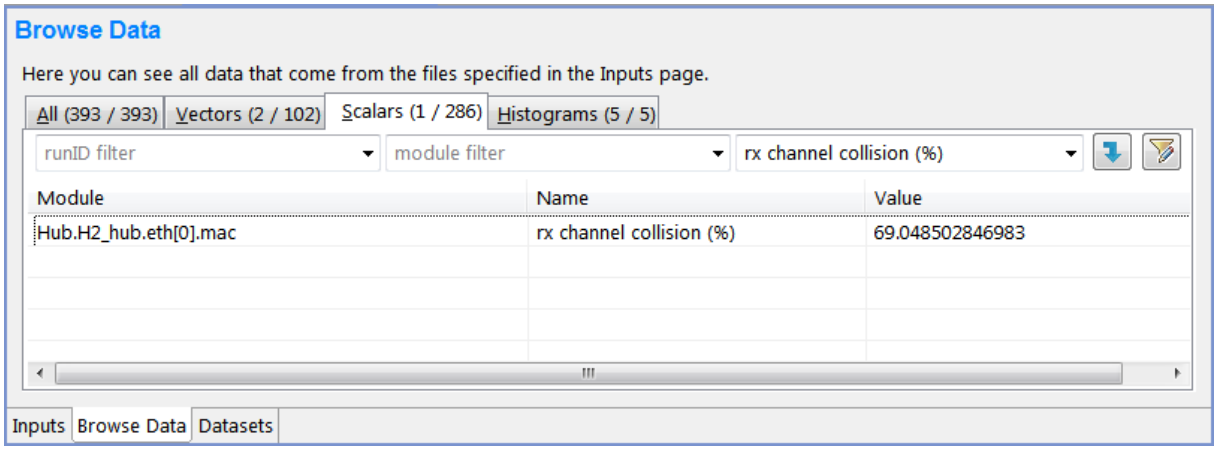
Vypracování 3. část – Hub - kolizní doména

Vytvoříme další tři scénáře s názvem [Config Hub_collisionsX], ve kterých budeme sledovat množství kolizí v síti s rozbočovači v závislosti na přibývajícím počtu stanic vysílajících data. Kolize vznikají, pokud na síťový prvek dorazí data ve shodnou dobu a v síti obsahující rozbočovače může velmi rychle přibývat výskyt kolizí, jelikož jsou jimi veškerá data šířena broadcastem. Na konec konfiguračního souboru **omnetpp.ini** vložíme následující kód:

```
[Config Hub_collisions_1]
network = Hub
**.H1_hub.tcpApp[0].connectAddress = "H4_hub"

[Config Hub_collisions_2]
extends = Hub_collisions_1
**.H4_hub.tcpApp[0].connectAddress = "H1_hub"

[Config Hub_collisions_3]
extends = Hub_collisions_2
**.H2_hub.numUdpApps = 1
**.H3_hub.numUdpApps = 1
**.H2_hub.udpApp[0].typename = "UDPBasicApp"
**.H2_hub.udpApp[0].destAddresses = "H3_hub"
**.H2*.udpApp[0].destPort = 2233
**.H2*.udpApp[0].startTime = 0.1s
**.H2*.udpApp[0].stopTime = 10s
**.H2*.udpApp[0].messageLength = 500B
**.H2*.udpApp[0].sendInterval = 0.1s
**.H3*.udpApp[0].typename = "UDPSink"
**.H3*.udpApp[0].localPort = 2233
```

Spustíme simulace a postupně vybereme všechny tři nově vytvořené konfigurace (Hub_collisions_1, Hub_collisions_2, Hub_collisions_3) K urychlení průběhu simulace můžeme využít ikonu **Express (5)**.

Po dokončení simulací se přesuneme do složky **results**, kde se vytvořili nové soubory s výsledky pojmenované dle názvů konfigurací, otevřeme pro každý z nich okno s výsledky (dvojklikem na jeden ze souborů pro každou konfiguraci). Přepneme se na záložku **Browse Data » Scalars**. Do pole **module filter** zadáme ***.H2_hub.***. V poli **statistic name filter** vybereme statistiku **rx channel collision (%)**, která zobrazuje procentuální poměr dat přenesených bez kolizí k celkovému množství přenášených dat (procento kolizí tedy lze získat odečtením naměřené hodnoty od sta procent).



Obrázek 6: Zobrazení procentuálního počtu kolizí pro scénář Hub_collisions_3

Otázky:

1. Na jaké vrstvě modelu ISO/OSI pracuje hub, switch a router?
2. Jaké informace obsahuje směrovací tabulka switche?
3. Jaké informace obsahuje směrovací tabulka routeru?