VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV INFORMAČNÍCH SYSTÉMŮ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF INFORMATION SYSTEMS

MODELOVÁNÍ A SIMULACE NÁVRHOVÝCH VZORŮ SMĚROVÁNÍ V POČÍTAČOVÝCH SÍTÍCH

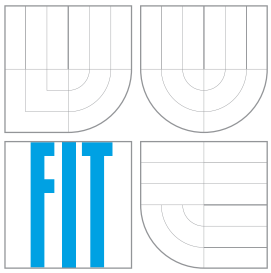
BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE
AUTHOR

VERONIKA RYBOVÁ

BRNO 2009

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV INFORMAČNÍCH SYSTÉMŮ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF INFORMATION SYSTEMS

MODELOVÁNÍ A SIMULACE NÁVRHOVÝCH VZORŮ SMĚROVÁNÍ V POČÍTAČOVÝCH SÍTÍCH

MODELLING AND SIMULATION OF NETWORK DESIGN GUIDES FOR IP ROUTING

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE
AUTHOR

VERONIKA RYBOVÁ

VEDOUCÍ PRÁCE
SUPERVISOR

Ing. PETR MATOUŠEK, Ph.D.

BRNO 2009

Abstrakt

Tato bakalářská práce se zabývá modelováním a simulací sítí v nástrojích Packet Tracer a OMNeT++. Modely sítí jsou vytvořeny podle návrhových vzorů firmy Cisco a zaměřují se na směrovací protokoly RIP, OSPF a redistribuci. Aby byly modely plně funkční, je pro OMNeT++ implementován protokol RIP a redistribuce z protokolu OSPF do protokolu RIP. Praktické využití nástrojů je ukázáno na simulaci dostupnosti a stability sítě. Práce zkoumá výsledky simulací a využití obou nástrojů pro simulaci sítí. Využíváme návrhové vzory určené pro návrh směrování v reálných sítí a implementujeme protokol podle standardu RFC. Proto předpokládáme praktické využití při analýze a simulací firemních a akademických počítačových sítí.

Abstract

This bachelor's thesis describes simulation of network using tools Packet Tracer and OMNeT++. Models of network are created according to design guides by Cisco company, which use routing protocols RIP, OSPF and redistribution. In order to provide full functionality of models in OMNeT++ it is necessary to implement protocol RIP and redistribution from protocol OSPF to protocol RIP. Practical usage of tools is demonstrated by simulations of accessibility and stability of network. Thesis investigates results of simulations and usage of both tools for simulation of networks. We use design guides, which are destined to design of real networks, and implement protocol according to standard RFC, therefore we suppose that this thesis will have practical usage for analysis and simulation of company's and academical networks.

Klíčová slova

Simulace sítí, modelování sítí, analýza sítí, RIP, OSPF, OMNeT++, Packet Tracer, Cisco

Keywords

Simulation of networks, modelling of networks, analysis of networks, RIP, OSPF, OMNeT++, Packet Tracer, Cisco

Citace

Veronika Rybová: Modelování a simulace návrhových vzorů směrování v počítačových sítích, bakalářská práce, Brno, FIT VUT v Brně, 2009

Modelování a simulace návrhových vzorů směrování v počítačových sítích

Prohlášení

Prohlašuji, že jsem tuto bakalářskou práci vypracovala samostatně pod vedením pana Ing. Matouška, Ph.D.

.....
Veronika Rybová
15. května 2009

Poděkování

Děkuji zejména svému vedoucímu Ing. Petru Matouškovi, PhD. za odbornou pomoc a stálou časovou dostupnost. Poděkování si zaslouží i další kolegové z ANSA týmu, kteří mi taktéž přispěli odbornými radami. Nakonec bych ráda poděkovala své rodině a příteli za psychickou podporu při zpracovávání bakalářské práce.

Obsah

Seznam obrázků

Úvod

Tato práce popisuje možnosti simulace sítí a to zejména za účelem zjištění chování sítě za různých okolností. Zaměřuje se především na simulaci směrovacích protokolů a to konkrétně RIP a OSPF, které se v dnešní době ve firmách nejčastěji využívají. Pro tyto účely využijeme simulační nástroje Packet Tracer a OMNeT++ a na vybraných vhodných topologiích předvedeme simulaci stavů, ke kterým v rámci směrování v sítích běžně dochází.

Rozhodneme-li se vytvořit zcela novou síť, např. pro vznikající firmu či její pobočku, může být náročné zjišťovat, jaké síťové prvky, jaké jejich nastavení a jaká topologie jsou pro ni nejvhodnější. Zakoupení různých druhů síťových prvků a následné experimentování s cílem dosáhnout, co nejvýhodnější topologie a konfigurace, je finančně i časově náročné.

Podobně zjišťování parametrů u již existující sítě není lehký úkol. Testovat takovou síť za chodu může způsobovat problémy, obzvláště pokud se rozhodneme zjišťovat, jak bude reagovat síť při pádu několika směrovačů při maximálním zatížení sítě. To sebou nese problémy nejen s uživateli této sítě, ale firmě může také přinést nechtěné finanční ztráty.

Jako nejlevnějším a nejefektivnějším řešení se jeví využití simulace sítě. V simulačním nástroji můžeme modelovat aktuální nebo zamýšlenou topologii sítě a podrobit ji sérii nejrůznějších testů. Dnešní simulační nástroje většinou podporují i grafické uživatelské rozhraní, díky kterému je možné vizuálně sledovat chování sítě.

Cíl práce

Tato práce by měla být uceleným návodem, jak provádět simulování sítí. Zaměří se na směrovací protokoly RIP a OSPF, které se využívají v sítích nejčastěji. Povede čtenáře od základních informací nutných k pochopení simulací sítě až k provedení vlastních simulacích. Nahlédne pod pokličku simulačních nástrojů Packet Tracer a OMNeT++, na kterých předvede, jaké rysy jsou pro simulaci podstatné. Popíše modelování několika vybraných topologií, navrhne, které aspekty je podstatné simulovat, a simulaci provede.

Struktura práce

První kapitola se zabývá tím, co je simulace sítí a jaké prostředky je možné využít.

V druhé kapitole se zabýváme směrováním a konkrétními směrovacími protokoly RIP a OSPF. Zmíníme se zde také o návrhových vzorech, které použijeme pro simulaci.

V třetí kapitole je popsána tvorba simulačních modelů na základě vybraných návrhových vzorů v nástrojích Packet Tracer a OMNeT++.

Čtvrtá kapitola se zabývá implementací chybějících modulů do nástroje OMNeT++.

V páté kapitole si předvedeme simulaci vybraných vlastností sítí v nástrojích Packet Tracer a OMNeT++ s využitím vytvořených simulačních modelů.

Šestá kapitola se zaměří na zhodnocení výsledků simulací z předchozí kapitoly a zhodnocení obou simulačních nástrojů. Nakonec shrneme možnosti praktického využití těchto simulací a simulačních nástrojů.

Kapitola 1

Simulační nástroje

Smyslem této práce je popsat simulace sítí, proto je podstatné si na začátku vysvětlit, co to vlastně simulace sítí je a co všechno zahrnuje. Následně popíšeme, jak se pracuje při simulace sítí v simulačních nástrojích Packet Tracer a OMNeT++. Na závěr oba tyto nástroje srovnám a zhodnotím jejich simulační možnosti.

1.1 Modelování a simulace sítí

V této sekci se kromě simulace jako takové zaměříme na popis diskrétní simulace, která se při simulacích sítí nejčastěji používá. Velká část této sekce se bude zabývat simulací zaměřenou již konkrétně na sítě.

1.1.1 Modelování a simulace obecně

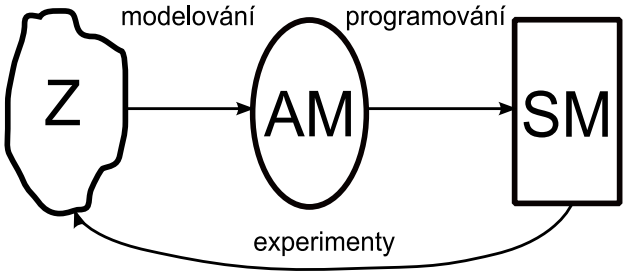
Modelování je proces vytváření modelu systému. Předpokládejme, že systém je soubor elementárních částí, které mezi sebou mají určité vazby. Model je určitou abstrakcí systému. Modelování je zásadní a podstatný krok, od kterého se vyvíjí výsledky simulace. Kvalita modelu je závislá na naších znalostech o systému. Simulace je experimentování s modelem systému za účelem získávání nových poznatků o modelovaném systému.

Tvorba modelu a jeho vhodný popis, stejně jako jeho správnost, jsou pro simulaci klíčové. Nejdříve je třeba systém dobře prozkoumat a ujasnit si, které jeho aspekty chceme modelovat. Po té vytvoříme abstraktní model, který neobsahuje všechny podrobnosti reálného systému, ale jen ty, o kterých jsme rozhodli, že jsou pro simulování podstatné. Tím dojde často k výraznému zjednodušení a model je pak snadnější implementovat.

Na základě abstraktního modelu vytvoříme model simulační. Mezi oběma modely je homomorfní vztah, takže už nedochází k žádnému zjednodušení. Simulační model má ale za úkol, narozdíl od abstraktního, sloužit k simulacím a experimentováním. Často je proto simulační model vlastně programem, který postihuje abstraktní model v celém rozsahu.

Následují již samotné simulační experimenty. V průběhu simulace sbíráme data. Jejich analýzou získáváme nové poznatky o systému a tím se celý simulační kruh uzavírá. V analyzační fázi se také zabýváme vhodnou prezentací výsledků v podobě grafů, histogramů apod. pro jejich lepší znázornění. Celý proces je znázorněn na obrázku 1.1. Podrobnější informace se můžete dozvědět v [11].

Simulace mohou být spojité, diskrétní nebo kombinované (obsahuje diskrétní i spojité části). Každá z nich se hodí na něco jiného. Při simulaci sítí se ale nejčastěji používá



Obrázek 1.1: Proces modelování a simulace (Z = znalosti; AM = abstraktní model; SM = simulační model).

diskrétní simulace, neboť jde o zasílání zpráv mezi síťovými prvky. Proto se v následující části budeme zabývat jen tímto typem.

1.1.2 Diskrétní simulace

Diskrétní simulace jsou založeny na změně stavu systému v určitém, předem naplánovaném čase. Stavy jsou měněny tzv. událostmi. Předpokládá se, že událost se vykoná v nulovém času, a mezi jednotlivými událostmi se nic neděje a systém se nijak nemění.

Podstatným pojmem pro každou simulaci je simulační čas. Ten se podstatně liší od času reálného nebo procesorového (jak dlouho trvá vykonání programu). Oproti času reálnému ho můžeme dle potřeby zrychlit nebo zpomalit. V diskrétních simulacích je se simulačním čase spojen kalendář událostí, který řídí průběh simulace.

Kalendář událostí je uspořádaná datová struktura, která uchovává záznamy o naplánovaných událostech a to tak, že jsou v kalendáři seřazeny dle času události od událostí s nejbližším časem vykonání. Simulátor pracuje ve smyčce. Pokud kalendář není prázdný, vybere první událost, nastaví simulační čas na dobu, kdy se má událost vykonat, a událost provede. Pokud jsou nějaké dvě události naplánované na stejný čas, rozhoduje o postupu zpracování jejich priorita. Pokud i ta je shodná, rozhodne náhodně simulátor.

1.1.3 Simulace sítí

Jak již bylo zmíněno v kapitole , pokud navrhujeme novou síť nebo rozšiřujeme síť stávající, může nám simulace velmi pomoci. Je vhodné navrhnout několik topologií, ty simulovat, experimentovat s různými problémy a pokusit se o nejlepší návrh. Zajímavou možností je využití emulace. Jde o propojení simulace s reálným provozem.

Simulace sítí se nejčastěji provádějí z důvodů abychom zjistili:

- Jaký dopad bude mít na síť větší vytíženost určitých linek či zařízení oproti ostatním.
- Jaký dopad na výkon bude mít změna topologie, rozšíření sítě nebo výměna některých síťových zařízení za novější.
- Jak se síť bude chovat, pokud dojde k chybě linky nebo některého síťového zařízení.
- Která zařízení nebo aplikace způsobují největší zpomalení sítě.
- Kolik uživatelů zvládne síť obsloužit bez větších zpoždění.

- Jak dlouho trvá síťovým aplikacím, než odpoví na požadavky uživatelů.

Existují také problémy, které jsou pomocí simulací téměř neřešitelné, jako je např. zjištění, zda jsou zařízení propojena korektně, optimalizace návrhu a výkonu sítě, simulování rozsáhlých sítích a v neposlední řadě simulace bezdrátových sítí, kdy je vliv okolního prostředí na tolik významný, že ho není možné zanedbat.

Typickými událostmi při diskrétních simulacích sítí je vyslání paketu, přijetí paketu a vypršení časovače (Timeout). Do kalendáře událostí jsou události vkládány podle času, kdy mají dorazit do svého cíle. Více o simulaci sítí se lze dočíst v [12].

Existuje velké množství simulačních nástrojů, které mají různé zaměření. Některé z nich vnikly pro výukové účely, např. Packet Tracer [24], CNET [19] nebo NetSim [28]. PARSEC [29], GloMoSim [18] nebo QualNet Developer [27] podporují paralelní simulaci. Pokud bychom rádi vyhodnotili výkon naší sítě, pak jsou vhodné nástroje Performance PROPHET [25] nebo QualNet Developer [27]. Na simulování bezdrátových sítí se zaměřili produkty GloMoSim a SWANS [22] postavený na JiST. Nástroj Traffic [20] se zabývá zpožděním v síti. Existuje ale i řada obecně zaměřených nástrojů jako je OMNeT++ [23], CNET, GTNetS [21] a další.

1.2 Nástroj Packet Tracer

V této části se budeme zabývat simulačním nástrojem od firmy Cisco – Packet Tracer (dále jen PT). Pro popis jsme zvolili nejnovější verzi PT 5.0 [24]. Zjišťovali jsme, jaké schopnosti má tento program v oblasti modelování a simulaci sítí. Návod na instalaci programu je v příloze B. Další informace můžete nastudovat v stručném návodu [3].

1.2.1 Popis prostředí a možností programu Packet Tracer

PT nabízí vizualizační a simulační nástroje, nástroje pro sestavení topologie z modelů reálných síťových prvků od firmy Cisco či nástroje pro vytváření nejrůznějších aktivit a testů pro studenty. Důraz se klade na množství síťových prvků této firmy, které si může uživatel sestavit, propojit a nastavit dle vlastního uvážení.

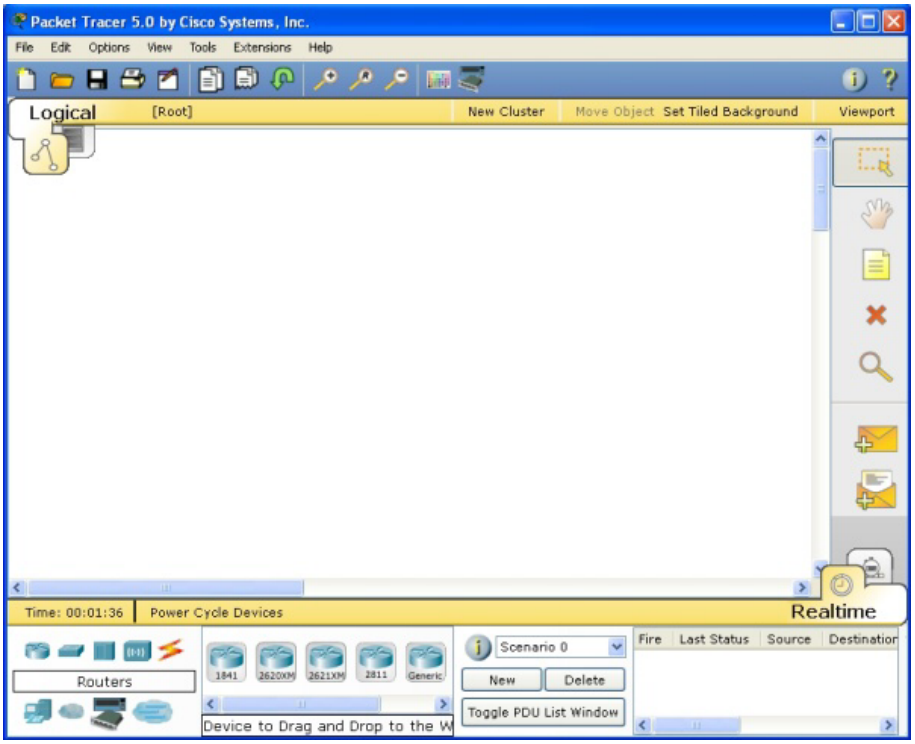
Vizualizace a animace jsou vhodné při simulaci reálného provozu sítě, díky nimž má uživatel možnost sledovat, v jakém čase, odkud kam putují jednotlivé PDU[1] (Protocol Data Unit), včetně toho jaké informace nesou. Packet Tracer podporuje protokoly HTTP, Telnet, SSH, TFTP, DHCP, TCP, UDP, IPv4, IPv6, ICMPv4, ICMPv6, RIP, EIGRP, OSPF, statické směrování, distribuce cest, Ethernet/802.3, 802.11, HDLC, Frame Relay, PPP, ARP, CDP, STP, RSTP, 801.1q, VTP, DTP a PAgP.

Prostředí programu po je zobrazeno na obrázku 1.2.

1.2.2 Modelování sítí

Pro modelování sítí obsahuje PT mnoho možností a vcelku věrně napodobuje skutečné vytváření sítí (kabeláž, konfigurace). Nabízí nepřeberné množství modelů síťových prvků a kabelů k jejich propojení. Můžeme vybírat z několika druhů směrovačů, přepínačů, rozbočovačů a bezdrátových zařízení od firmy Cisco. Podobně u kabelů si je možné vybrat, zda použijeme křížený, přímý, sériový, koaxiální či optický kabel.

[1] Obecný název pro datovou jednotku zasílanou po síti.



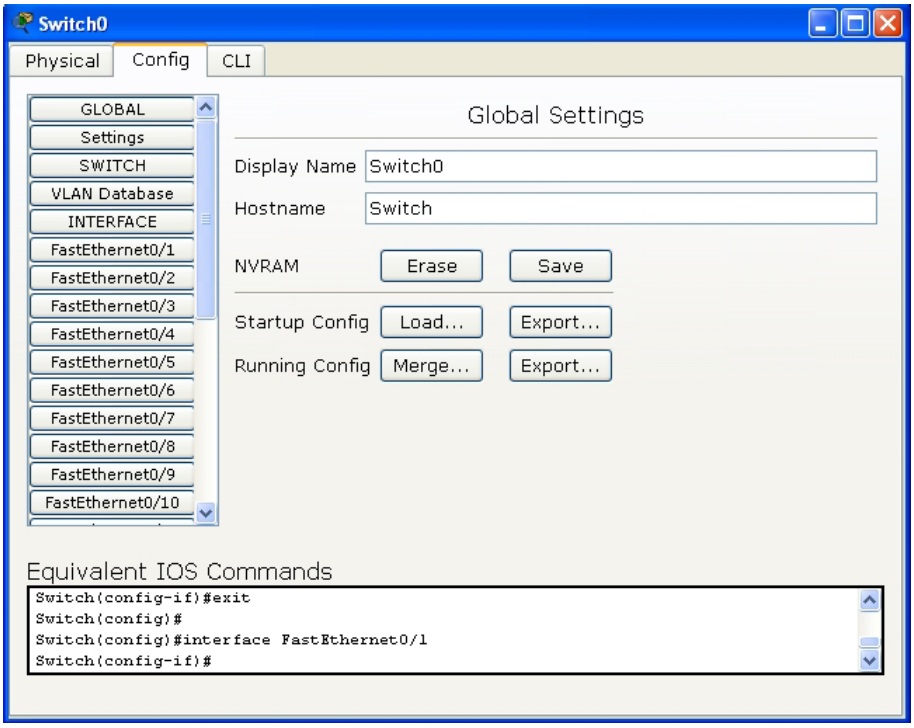
Obrázek 1.2: Prostředí programu PT.

Většina z nabízených zařízení je modulárních a tak i v PT je možné přidávat do zařízení vybrané moduly. Například základní směrovač Cisco 1841 umožňuje přidání síťových karet s ethernetovými porty (RJ-45), telefonními porty (RJ-11) nebo sériovými porty. Všechna zařízení je možné konfigurovat tak, jako by se jednalo o zařízení reálná. Směrovače a přepínače lze nastavovat pomocí příkazů operačního systému IOS. Zde však musíme podotknout, že PT neumožňuje používat příkazy IOS v plné šíři, ale běžně užívané příkazy zvládne.

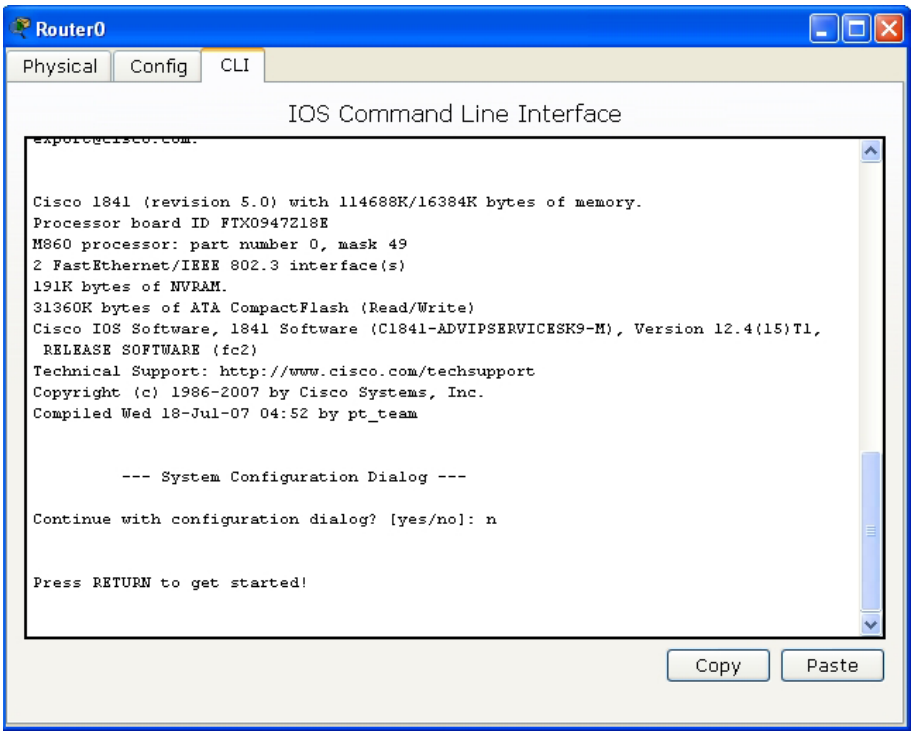
Modelování sítí se provádí v logickém pracovním prostotu. Do něj vkládáme síťové prvky z nabídky vlevo dole. Pro propojení zařízení nejprve vybereme potřebný druh kabelů z nabídky. Kliknutím na zařízení a výběrem rozhraní, do kterého kabel zapojíme, vybereme první zařízení. Podobně to provedeme s druhým zařízením, které chceme k prvnímu kabelem připojit.

Můžeme pokračovat nastavením jednotlivých zařízení. Klepnutím na zařízení se dostaneme do jeho nastavení. Je možné pracovat přes uživatelské rozhraní v záložce Config (obrázek 1.3), které je vhodné především pro začátečníky nebo běžné uživatele. U přepínače a směrovače můžeme využít možnosti profesionálního nastavení zařízení v záložce CLI (obrázek 1.4), kde se simuluje připojení k zařízení přes konzolové rozhraní a umožňuje využít operační systému IOS.

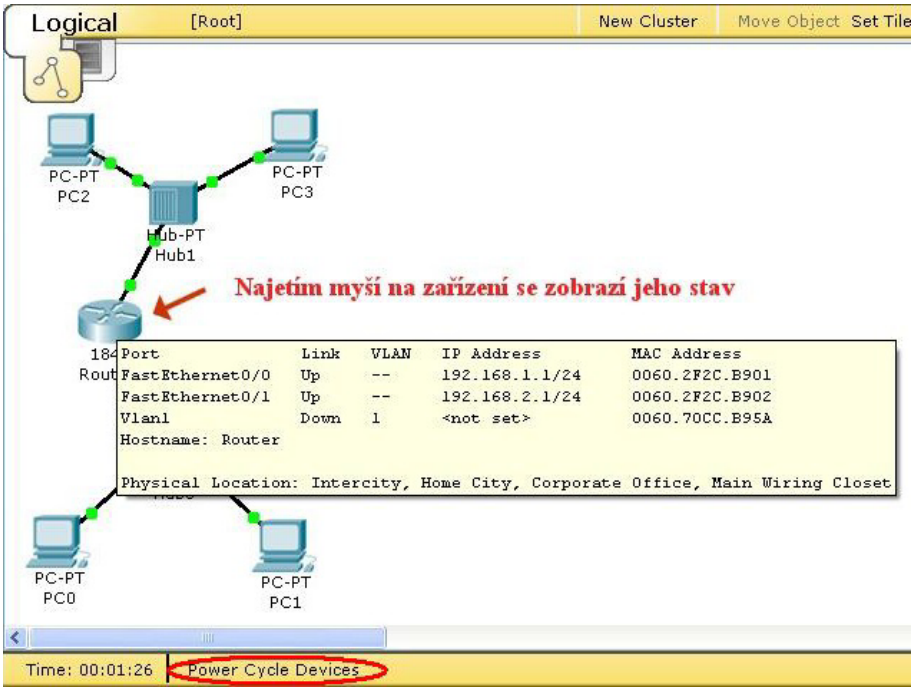
Zjištění některých základních nastavení zařízení lze provést pouhým umístěním kurzoru na dané zařízení. Na obrázku 1.5 můžeme vidět, že se objeví nápověda obsahující nastavení jednotlivých rozhraní zařízení (Up/Down, VLAN, MAC a IP adresa). V Realtime módu lze všechna zařízení restartovat použitím tlačítka Power Cycle Devices vyznačeného v obrázku 1.5 červeným oválem. To je praktické chceme-li sledovat chování protokolu od momentu jeho spuštění na zařízení.



Obrázek 1.3: Konfigurační rozhraní přepínače v PT.



Obrázek 1.4: Rozhraní s operačním systémem IOS v PT.



Obrázek 1.5: Základní informace o zařízení v PT.

1.2.3 Simulace sítí

Simulační mód umožňuje upravit časovou osu, sledovat zprávy, které si v síti zařízení zasílají, sledovat jejich trasy, zkoumat podrobně jejich obsah a vysílat vlastní zprávy. Simulace v PT je diskrétní, tedy další akce v čase se provádějí skokově.

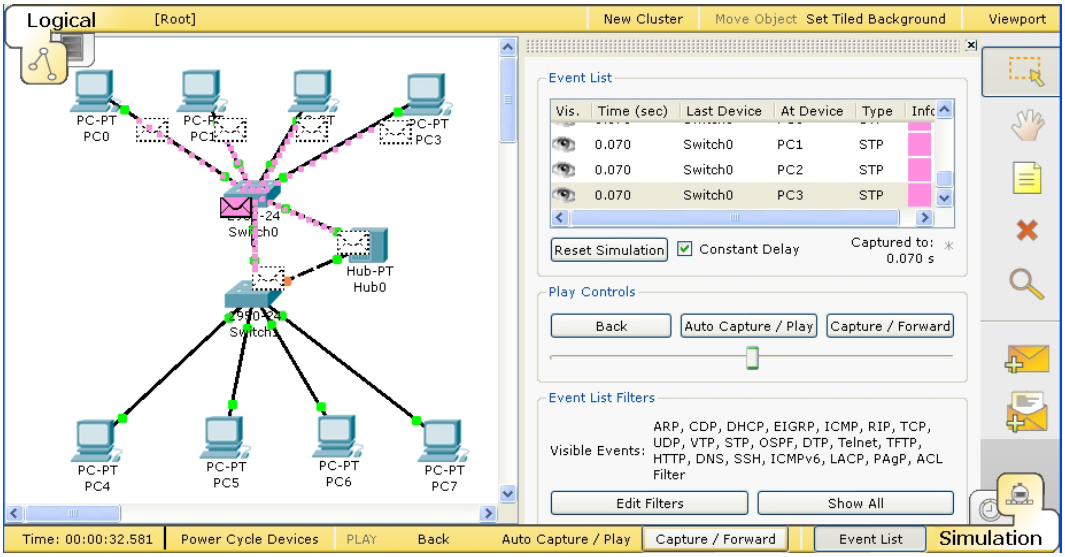
Popis simulačního módu

V simulačním módu je pro simulaci připraven simulační panel. Na obrázku 1.6 vidíme, že se skládá ze tří částí. V první části s názvem Event List se při simulaci bude zobrazovat záznam o posílaných paketech. Ke každému PDU se zde zaznamenává čas, ve který byl zaslán, z jakého zařízení byl zaslán, na kterém zařízení se právě nachází a protokol, který využívá. Pro lepší orientaci ještě obsahuje dodatkové informace o tom, jestli je PDU právě viditelné a jakou barvu má. PDU jsou totiž v topologii zobrazovány jako obálky různých barev (obrázek 1.6).

Další část s názvem Play Controls umožňuje simulaci ovládat. Pomocí tlačítka Auto Capture/Play můžeme nechat simulaci plynule běžet. Pro konkrétní řešení nějakého problému jsou pak vhodnější tlačítka Capture/Forward a Back, které provádí v simulaci jeden krok dopředu nebo vzad. Na posuvníku můžeme určovat jak rychle se bude simulace provádět.

V poslední části Event List Filters můžeme filtrovat provoz, který je v simulaci zobrazován. Implicitně jsou povoleny všechny protokoly, které PT umí. Pomocí tlačítka Edit Filters lze určit, které protokoly budeme sledovat, a nastavit ACL filtr.

Resetováním simulace se simulační čas nastaví na 0,000 sekund a vymaže se EventList. Resetování simulace provedeme tlačítkem Reset Simulation, restartováním zařízení (tlačítkem Power Cycle Devices) či modifikováním topologie sítě.



Obrázek 1.6: Simulační mód.

Studium PDU

Každou jednotku PDU, která se zobrazí v Event Listu, můžeme důkladně prozkoumat. PT má struktury PDU velmi dobře graficky zpracované a tak se v nich lehce vyzná i začátečník. Po informativní stránce jsou plnohodnotné, takže i odborník v nich najde vše potřebné. PDU informace získáme kliknutím na obálku v topologii nebo na záznam v Event Listu.

V záložce OSI Model (obrázek 1.7) je názorně zobrazeno, jak je zpráva zpracovávána jednotlivými vrstvami OSI modelu. Pod obrázkem znázorňujícím OSI model je postup detailně popsán.

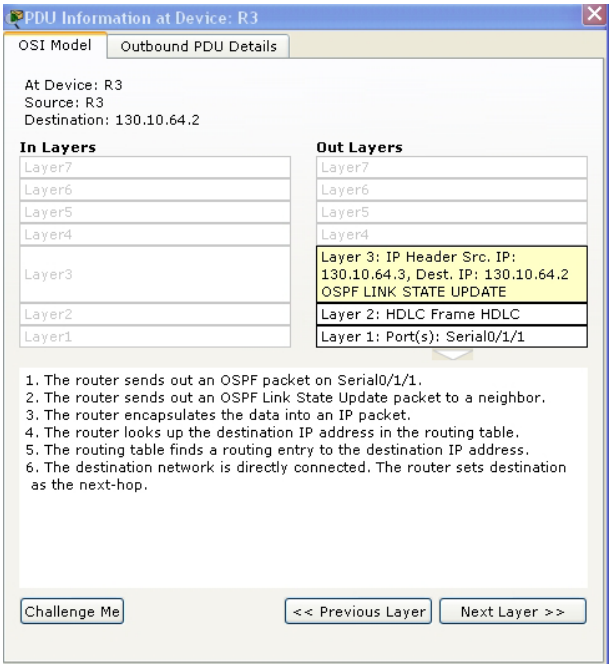
Záložka PDU Details (obrázek 1.8) zobrazuje obsah PDU. Ten je graficky organizován do tabulek znázorňujících různé typy hlaviček, které PDU obsahuje. Ty jsou dále členěny na pole obsahující informace.

1.3 Nástroj OMNeT++

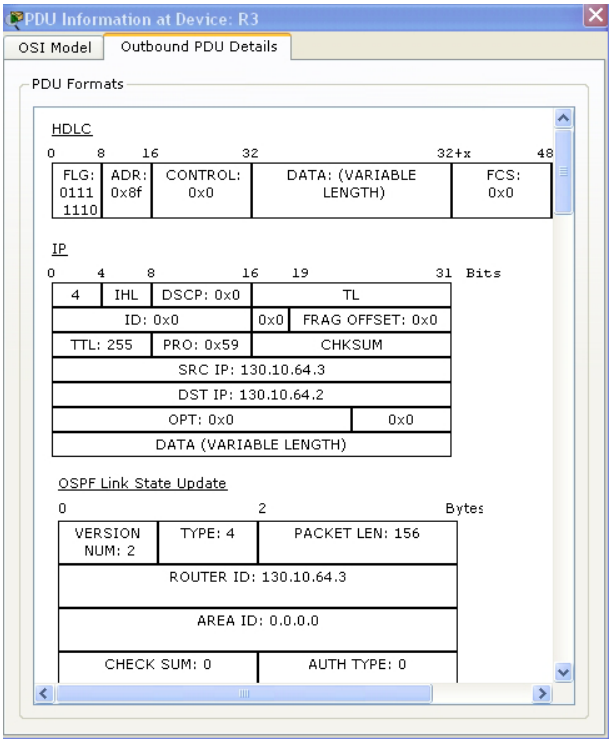
Vzhledem k tomu, že se v této práci máme zabývat možnostmi simulačního nástroje OMNeT++, bude tato sekce věnována právě jemu. OMNeT++ je pro tuto práci vhodný, protože je volně dostupný a má otevřený zdrojový kód. Jeho další výhody budou uvedeny dále. Pro popis jsme zvolili poslední verzi 4.0. Všechny informace byly čerpány z velmi dobře zpracovaného manuálu [17]. Stejně jako u PT jsme zjišťovali, jaké schopnosti má tento program v oblasti modelování a simulaci sítí.

1.3.1 Možnosti programu OMNeT++

OMNeT++ je objektově orientovaný modulární diskrétní síťový simulátor. Je založený na objektovém jazyku C++ a vlastním jazyku NED. Existují verze pro Unix i Windows a obě verze jsou pro školní účely zdarma. Kromě simulačního jádra obsahuje GUI a IDE. Obsahuje knihovny pro práci s TCP/IP, Ethernet, FDDI, Token Ring, 802.11 a Peer-to-peer.



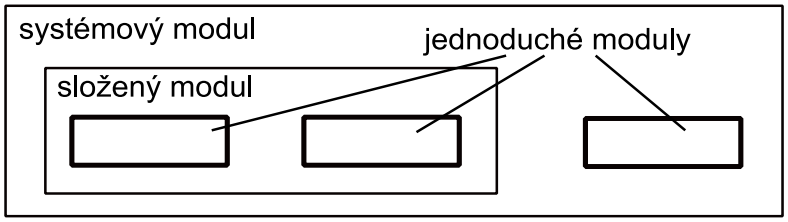
Obrázek 1.7: OSI model PDU v PT.



Obrázek 1.8: Struktura PDU v PT.

1.3.2 Modelování

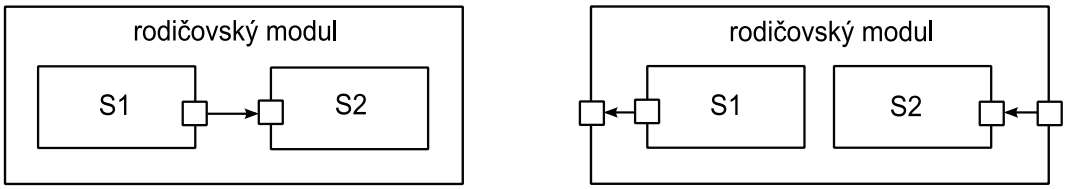
Při modelování v OMNeT++ se do sebe vnořují jednotlivé moduly hierarchicky. Nejvýše postavený modul se označuje jako modul systémový, ten se skládá ze submodulů (obrázek 1.9). Může se jednat o modely složené, které se skládají z dalších složených modulů nebo z modulů jednoduchých. Ty jsou dále nedělitelné.



Obrázek 1.9: Hierarchická struktura modulů.

Topologie sítě – propojení jednotlivých modulů – se v OMNeT++ popisuje speciálním jazykem NED. Jednoduché moduly obsahují algoritmus, který se zapisuje v jazyce C++. Moduly není třeba psát od začátku. OMNeT++ obsahuje několik předdefinovaných tříd objektů, u kterých je třeba jen redefinovat chování.

Moduly mezi sebou komunikují pomoci zasílání zpráv. Ty budeme většinou považovat za model PDU. Zprávy mohou přijít od jiného modulu nebo ze stejného (využívají se jako časovače). Každý modul obsahuje brány, které jsou vstupní (In) pro příjem zpráv a výstupní (Out) pro odesílání zpráv. Mezi bránami modulů se vytváří spojení. Spojení může existovat mezi moduly na stejné úrovni hierarchie nebo mezi modulem a jeho složeným rodičovským modulem (obrázek 1.10). Pro možnost modelování přenosu paketů po lince, má spojení tři volitelné parametry – přenosové zpoždění, bitová chybovost a rychlost přenosu dat.



Obrázek 1.10: Spojení submodulů mezi sebou a propojení rodičovského modulu se submoduly.

1.3.3 Jazyk NED

Jazyk NED slouží pro popis topologii sítě. Soubory obsahující popis jazykem NED musí mít příponu `.ned`. Způsob, kterým se popisují moduly, je rozdílný pro moduly složené a jednoduché. Popis jednoduchého modelu obsahuje kromě povinného názvu volitelně parametry a brány. Brány můžeme zadat pomocí vektoru.

```
simple SimpleModuleName          ; název jednoduchého modulu
{
   parametres:                   ; deklarace parametrů
```

```
        const example1;
        string example2;
    gates:                          ; deklarace bran
        input fromPort, input[];    ; vstupní brány (pomocí skaláru i vektoru)
        output toPort, output[];    ; výstupní brány
}
```

Popis složeného modulu může volitelně obsahovat parametry, brány, submoduly a popis spojení. Povinný je jen název. Parametry a brány mají stejný význam jako u jednoduchých modulů. Submoduly jsou moduly, ze kterých se složený modul skládá. Spojení jsou klíčová pro vytvoření topologie a udávají, která vstupní brána se propojí s kterou výstupní bránou.

```
module CompoundModuleName               ; název složeného modulu
{
    parametres:                         ; deklarace parametrů
        const example1;
        string example2;
    gates:                              ; deklarace bran
        input fromPort;
        output toPort;
    submodules:                         ; deklarace modulu (typ a název)
        node1: Node {};
        node2: Node {};
    connections:                        ; definice propojení bran
        node1.output --> node2.input;
        node1.input <-- node2.output;
}
```

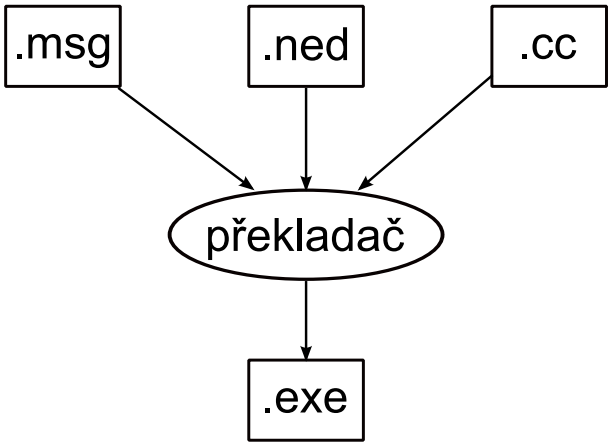
Soubory `.ned` lze zapisovat textově nebo pomocí jednoduchého uživatelsky přívětivého nástroje GNED, který OMNeT++ nabízí.

1.3.4 Simulace

Poté, co máme napsané všechny topologické popisy v souborech s příponou `.ned`, popisy zpráv v souborech s příponou `.msg` a popisy chování jednoduchých modulů v jazyků C++ v souborech s příponou `.cc`, můžeme celý simulační program sestavit. Výsledkem je spustitelný soubor `.exe` (pod Windows), který již nepotřebuje knihovnu OMNeT++ a je spustitelný na libovolném PC (obrázek 1.11).

Pro nastavení parametrů simulace ještě budeme potřebovat soubor `omnetpp.ini`, jinak bychom museli toto nastavení dělat při každém spuštění simulace ručně. Tento soubor má vlastní specifickou syntax, která je dopodrobna popsána v [23].

K dispozici jsou dva druhy uživatelských rozhraní – příkazová řádka (Cmdenv) a grafické (Tkenv založen na Tcl/Tk). Pro větší názornost simulace je vhodný grafický režim, pokud nás ale zajímají pouze výsledky simulace, postačí nám příkazová řádka. V grafickém režimu můžeme sledovat, jak zprávy putují mezi zařízeními, zrychlovat/zpomalovat simulaci, sledovat grafické výstupy statistik (histogramy atd.), restartovat animaci a další. Bohužel simulaci nejde vracet v čase.



Obrázek 1.11: Diagram popisující vytvoření spustitelného souboru v OMNeT++.

Simulační prostředí můžeme vidět na obrázku 1.12. Pod panelem nástrojů se nachází časová osa, na které jsou vyznačena události uložené v kalendáři. Vpravo na obrázku můžeme vidět modul představující topologii sítě a vlevo modul představující směrovač. Zprávy jsou symbolizovány červenou tečkou s názvem zprávy, které se pohybují topologií. Vpravo dole je okno s bližšími informacemi o zprávě. V případě paketu se zde můžeme dočíst všechny podstatné informace - MAC adresy, IP adresy, porty, obsah zprávy apod.

OMNeT++ je pro simulaci vybaven rozsáhlou simulační knihovnou, který skýtá mnoho možností. Existuje zde třída představující zasílané zprávy (`cMessage`), třída pro přístup k branám a parametrům zadaným v `.ned` souboru(`cModule`), funkce pro zasílání, přijímání zpráv a plánování událostí (`send()`, `schedulaAt()`, `endSimulation()`), funkce pro přístup k ostatním modulům (`parentModulu()`, `ownerModule()`, `toGate()`).

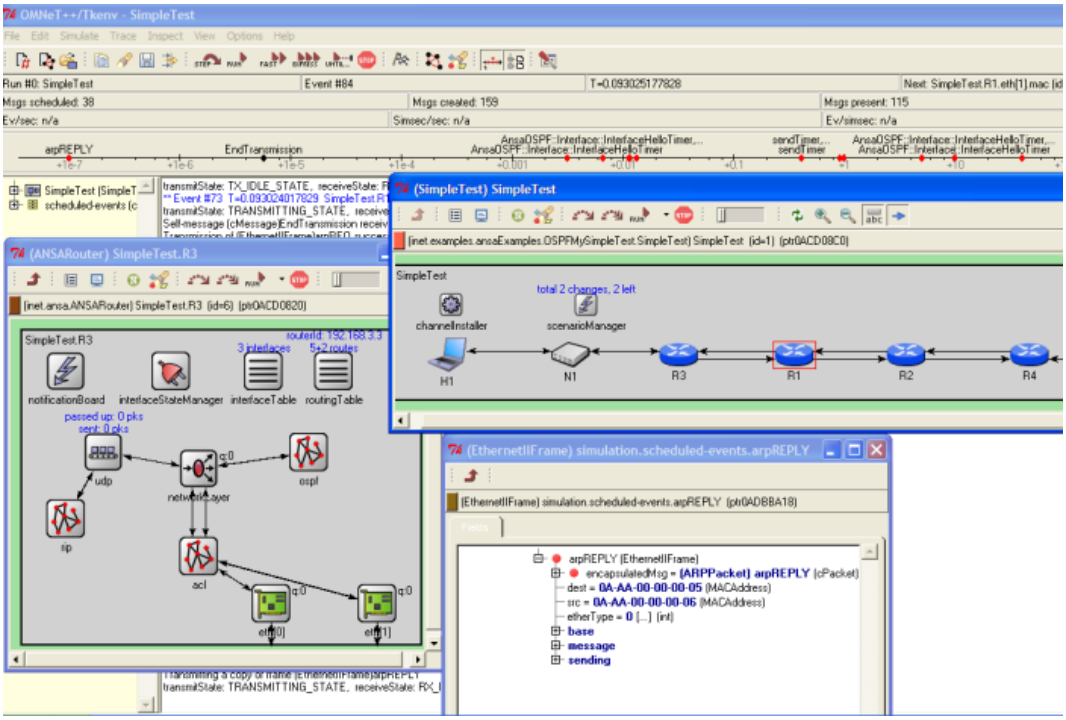
Statistická knihovna umožňuje zpracovávat výsledky a vypisovat počet vzorků, průměrné hodnoty, odchylky, minima, maxima. Má také několik možností implementace histogramů. Součástí OMNeT++ jsou nástroje Plove a Scalars, které mohou být použity pro zpracování a analýzu výsledků ve formě grafů. Plove pracuje s hodnotami v čase a Scalars s hodnotami skalárními.

1.4 Shrnutí kapitoly

Tato kapitola popsala, co to je simulace. Blíže popsala simulaci diskrétní a nakonec se zaměřila na simulování sítí. Byly zde také zmíněny simulační nástroje, které můžeme k simulaci použít. Dva z nich – Packet Tracer a OMNeT++ – byly následně blíže popsány. Oba nástroje jsou vhodné pro simulaci sítí, ale každý má své výhody i nevýhody.

OMNeT++ je univerzální simulační nástroj s velkými možnostmi. Pro nekomerční využití je zcela zdarma a je možné jej dále rozšiřovat o knihovny, které potřebujeme ke své práci. Má propracovaný jazyk pro popis topologie sítí a předpřipravené třídy pro popis nejjednodušších stavebních prvků topologie.

Oproti tomu PT mohou využívat pouze studenti a učitelé Cisco akademie a jeho kód není volně dostupný. Také všechna zařízení, ze kterých je možné vytvářet sítě, jsou pouze značky Cisco, ani ty zde však nejsou zastoupeny všechny. Program PT není možné prakticky nijak rozšířit. Není možné doplňovat další zařízení a kabely, upravovat jejich vlastnosti, ani



Obrázek 1.12: Simulační prostředí v OMNeT++.

si doprogramovat chybějící protokoly nebo jejich novější verze.

K modelování se v PT nabízí široká paleta síťových komponent. Jejich propojování je jednoduché a intuitivní. Taktéž nastavování jednotlivých síťových prvků je jednoduché jak pro začátečníka, tak pro profesionála, který může využít operační systém IOS. V OMNeT++ je třeba mnoho komponent dopsat, proto je v oblasti modelování sítí PT jednodušší a po mnoha stránkách lepší (reálné síťové prvky, IOS).

Simulace sítí je v PT velmi názorná a snadná. Zkoumání obsahů PDU je jednoduché, srozumitelné a přehledné. Simulace je ale poněkud primitivní a pro naše potřeby nejsou zcela dostačující. Neumožňuje simulovat některé základní akce jako je pád linky či porucha zařízení. Výsledky simulace není možné nijak vyexportovat. Je tedy vhodnější pro školní výuku než pro simulaci větší sítě. Simulace v OMNeT++ mohou být v grafickém režimu také velmi názorné. OMNeT++ je ale naproti PT ryzí simulační nástroj a tak nabízí velké množství tříd vhodných pro simulaci. OMNeT++ je tedy po simulační stránce jednoznačně lépe vybaven.

Díky tomu, že se jedná o univerzální nástroj, neobsahuje samotný OMNeT++ mnoho knihoven, které by byly vhodné pro simulaci sítě, předně zde chybí knihovny pro směrovaní (RIP i OSPF), které budeme dále potřebovat. Tento problém se ale dá řešit použitím framework INET, který pracuje nad OMNeT++ a nabízí knihovny pro simulace sítí. Ale ani v tomto případě nejsou knihovny dostačující a bude třeba si některé dopsat. Tento problém nemusíme u PT vůbec řešit, podporuje totiž všechny běžné směrovací protokoly.

Program PT byl určen ke studiu sítí. Jeho ovládání je jednoduché a intuitivní i pro laika. Pokud chce uživatel pracovat s OMNeT++, musí umět alespoň základy C++ a předpokládá se, že se naučí i nový jazyk NED. Nástroj je pro začátečníka poměrně složitý a zabere dost času, než se v něm zorientuje. Naštěstí manuál [23] je velmi dobře zpracován.

Kapitola 2

Návrhové vzory se zaměřením na protokoly RIP a OSPF

Tato kapitola se zabývá předmětem simulování. Budeme simulovat směrovací protokoly RIP a OSPF. Proto se krátce zmíním o tom, co to vlastně směrovaní je a popíšu zmíněné směrovací protokoly. Prozkoumám návrhové vzory od firmy Cisco a vyberu několik z nich, které budou pro simulaci vhodné.

2.1 Směrování

Tato kapitola je úvodem do směrování. Uvádíme zde základní informace o směrování a směrovací tabulce. Lehce se zmíníme o statickém směrování, dynamickém směrování a redistribuci. Čerpali jsme z [2] a [7].

2.1.1 Směrování a směrovací tabulka

Směrování je proces, při kterém směrovač zjišťuje cestu do cílové sítě. Tento proces probíhá na třetí vrstvě modelu ISO/OSI. K směrování potřebuje směrovač mít směrovací tabulku. V té se nachází všechny sítě, které zná a rozhraní, na které má poslat paket, pokud směruje některé z těchto sítí. Kam směřuje, zjistí směrovač z IP hlavičky paketu, přesně z cílové IP adresy. Informace o sítích se do tabulky mohou dostat různým způsobem. Záleží na tom, zda směrovač využívá statického, dynamického směrování či obou způsobů.

2.1.2 Statické směrování

Statické směrování funguje na základě statických záznamů v směrovací tabulce. Tyto záznamy se musejí do tabulky zadávat ručně a zvlášť pro každou síť, což může být náročné pro velké množství okolních sítích, a tak se často v takovémto případě nahrazuje směrováním dynamickým. Tyto záznamy se také mohou stát časem neaktuální a vyžadují znovu zásah administrátora.

Nejčastěji se proto používají pro zápis, tzv. Default Gateway. Jde o záznam v tabulce, který je vybrán vždy, pokud se jiný záznam v tabulce neshoduje s cílovou adresou sítě paketu. Výhodou staticky zadaných záznamů je to, že mají nejnižší administrativní vzdálenost ze všech možných způsobů směrování. Administrativní vzdálenost (Administrative Distance) udává důvěryhodnost zdroje cesty v tabulce. Záznamy zadané staticky ji

mají implicitně nastavenou na jedna. To znamená, že tyto záznamy jsou považovány za nejdůvěryhodnější a mají prioritu nad ostatními záznamy.

2.1.3 Dynamické směrování

Dynamické směrování, narozdíl od statického, umí reagovat na změny v topologii sítě. Pro vytváření záznamů v tabulce využívá algoritmy, které dokážou z informací, které přicházejí od sousedních směrovačů, vypočítat tu nejlepší cestu k cíli.

Existují různé způsoby dynamického směrování, jejichž chování, funkčnost a výměna informací mezi směrovači je podložena různými druhy směrovacích protokolů. Podstatné ale je, že vybraný protokol se nastaví na směrovači a dále již vše funguje bez zásahu administrátora. To znamená, že směrovače si sami zjistí okolní sítě a dokážou reagovat na změny topologie.

Sítě bývají rozděleny pro účely dynamického směrování do autonomních oblastí. Důvodem je, aby se informace o směrovacích tabulkách nešířily celým internetem a správa sítí byla snazší.

Podle druhu algoritmů, které směrovací protokoly používají, je dělíme na protokoly vektorově orientované (Distance Vector Routing Protocol) a protokoly stavu linky (Link-state). Vektorově orientované optimalizují cestu pouze podle vzdálenosti od zdroje k cíli. Směrovač si uchovává informaci o směru cesty jako vektor směru (Next hop) a vzdálenosti (Metric). Mezi takovéto protokoly patří RIP [5] a IGRP [6]. Link-state protokoly jsou typické tím, že si udržují kompletní mapu topologie sítě v tzv. topologické databázi (Link State Database). Takovými protokoly jsou OSPF [9] a IS-IS [10].

2.1.4 Redistribuce

V momentě, kdy různé části sítě využívají různé směrovací protokoly (v našem případě RIP a OSPF), je třeba využít redistribuci. Redistribuce je proces v rámci jednoho směrovače, který umožňuje překládat směrovací informace z jednoho směrovacího protokolu na druhý. Možná vás napadne, proč používat rozdílné směrovací protokoly.

Protokol RIP je nejstarším dnes užívaným protokolem, proto ho podporují snad všechny síťové prvky umožňující provádět dynamické směrování. Jak se dozvíme v následující kapitole, nemusí RIP vyhovovat požadavkům administrátorů a ti pak mohou sáhnout po některém z novějších protokolů jako je např. OSPF. Proto mohou v méně kritických částech sítě ponechat stará nebo méně dokonalá zařízení, která budou využívat protokol RIP a například páteřní část sítě vybavit prvky, které podporují OSPF protokol.

Hlavním problémem při směrovaní jsou různé metriky, které směrovací protokoly využívají. Hraniční směrovače (směrovače, které pracují s dvěma či více druhy směrovacích protokolů) se pak musí nastavit tak, aby dokázaly cesty propagovat s vhodnou metrikou.

2.2 Protokol RIP

Prvním protokolem, kterým se budeme zabývat je Routing Information Protocol (dále jen RIP). Je jedním ze směrovacích protokolů, jejichž chování máme simulovat. Bude uvedeno, jak RIP funguje, jakou metriku využívá a jak vypadají zprávy, které si směrovače mezi sebou zasílají. Taktéž se zmíníme jeho klady a zápory, které na závěr zhodnotíme.

2.2.1 Stručný popis

RIP je nejstarším a nejpoužívanějším směrovacím protokolem. Vznikl koncem 70. let a v roce 1988 se dočkal standardizace v podobě RFC 1058 [5]. Protokol RIP vděčí za svou oblíbenost především jednoduchosti, která předčí i jeho některé nedostatky. Většina z nich pak byla odstraněna v druhé verzi tohoto protokolu. My se zaměřím především na popis protokol RIP verze 1 (RIPv1), protože především ten budeme používat při této práci. Jak už bylo uvedeno výše, RIP patří mezi vektorově orientované protokoly.

2.2.2 Funkce protokolu RIP

RIP používá dva druhy zpráv – žádost (Request) a odpověď (Response) – a vždy zasílá všechny zprávy pomocí broadcastu. Po restartu směrovače rozešle směrovač na všechna svá rozhraní žádost (Request). Jeho sousedé mu odpoví pomocí zprávy (Response) obsahující jejich směrovací tabulky. Následně, pokud nedojde ke změně topologie, už směrovač posílá jen aktualizační zprávy (Update Message).

Když směrovač přijme zprávu obsahující směrovací tabulku od souseda, která obsahuje změny, provede aktualizaci své tabulky. Metrika pro cestu se zvýší o jedna a zapíše do tabulky. Podobně pokud přijde informace o cestě, kterou už směrovač v tabulce má, ale s lepší metriku, aktuální záznam přepíše tímto novým. Směrovač udržuje vždy jen informaci o té nejlepší cestě, tedy cestu s nejnižší metrikou. Soused, od kterého zprávu dostal, je označen jako směr cesty (Next Hop). Po aktualizaci vlastní tabulky, začne směrovač vysílat vlastní aktualizační zprávy, aby informoval o změnách sousední směrovače.

2.2.3 Metrika a administrativní vzdálenost

Protokol RIP používá jako metriku pro měření vzdálenosti mezi směrovačem a cílovou sítí počet skoků k cíly (Hop Count). Cesta mezi dvěma sousedícími směrovači je většinou ohodnocena jedničkou. Pokud tedy směrovač zasílá dvou směrovací tabulku svému sousedovi, před odesláním zprávy navýší všem cestám Hop Count o jedna. Jak již bylo zmíněno výše, IP adresa souseda je pak použita jako cíl pro nejbližší další skok (Next Hop).

RIP má implicitní administrativní vzdálenost 120, což znamená, že je jeden z nejméně důvěryhodných směrovacích protokolů. Například statická cesta má administrativní vzdálenost jedna, protokol EIGRP 90 nebo OSPF 110. V praxi to znamená, že pokud směrovací tabulka obsahuje stejné cesty z různých zdrojů, pak je pro směrování vybrána cesta s nejnižší administrativní vzdálenosti.

2.2.4 Třídní protokol

Směrovače používající protokol RIP zasílají pouze IP adresy cíle, ne masku. RIP je totiž třídní (Classful) protokol. Směrovač pak použije masku nakonfigurovanou na lokálním rozhraní nebo implicitní masku na základě třídy IP adresy. Toto může být, spolu s automatickou sumarizací, závažný problém.

Uveďme si příklad. Budeme-li uvažovat tři budovy firmy, každá bude mít vlastní IP adresu sítě, řekněme 10.10.0.0/16, 10.11.0.0/16 a 10.12.0.0/16. Směrovače v jednotlivých budovách budou propojeny sítí s IP adresou 192.168.1.0/24. V tomto případě nelze použít protokol RIP, neboť směrovače si budou posílat informace pouze o sítích s IP adresou 10.0.0.0/8 (dojde k sumarizaci podle typu třídy) a tedy nebude možné rozeznat jednotlivé firemní podsítě.

Ze stejného důvodu není možné RIP použít v prostředí, kde se pracuje s maskami proměnné délky (VLSM) nebo se směrováním na bázi adresových prefixů (CIDR).

2.2.5 Časovače

Pro lepší řízení výkonnosti protokolu, používá RIP tři časovače. Časovač Routing-update určuje časový interval mezi zasíláním periodických aktualizačních zpráv. Implicitně je nastaven na 30 sekund. Po restartu se k nim přičítá ještě malá prodleva, aby nedošlo k ucpání linky.

Časovač Route-timeout udává čas, do jehož uplynutí musí směrovač zaslat aktualizační zprávu, jinak bude cesta k němu považována za neplatnou. Tento časovač je implicitně nastaven na 180 sekund.

Následně odpočítává časovač Route-flush, po jehož uplynutí je neplatná cesta z směrovací tabulky vymazána. Časovač je implicitně nastaven na 240 sekund.

2.2.6 Stabilita protokolu

I když Bellman-Fordův algoritmus nedokáže předejít vznikům smyček, RIP má opatření k tomu, aby se to nestávalo. V protokolu je zaneseno, že maximální počet skoků (Next Hop) je 15. Pokud má nějaká cesta metriku 16, což je u RIP považováno za nekonečno, směrovač předpokládá, že cíl je nedostupný. Na druhou stranu je z tohoto postupu jasně viditelná nevýhoda. RIP má omezení na 15 skoků a tudíž nemůže být použit v rozsáhlých sítích.

Krom tohoto opatření má RIP i další vlastnosti, které jsou navržené pro zajištění stability při rychlých změnách topologie, např. Split Horizon. Ten zabraňuje tomu, aby směrovač zasílal cestu na rozhraní, skrz které se tuto informaci dozvěděl. Je tedy jednou z metod, která zabraňuje směrovacím smyčkám.

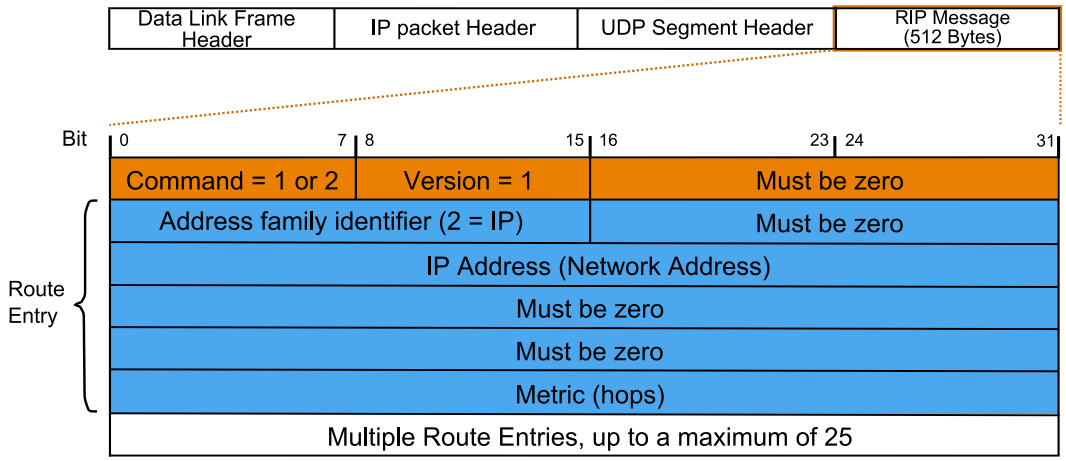
2.2.7 Formát zprávy

Protokol RIP pracuje na aplikační vrstvě. Na obrázku 2.1 je znázorněna struktura zprávy včetně její souvislost s nižšími vrstvami. RIP využívá jako transportní vrstvu UDP a má rezervovaný UDP port 520. Jak už bylo řečeno dříve, RIP využívá pro zasílání zpráv broadcast. Z toho vyplývá, že v IP hlavičce je zadaná cílová IP adresa 255.255.255.255 a v hlavičce síťového rozhraní je cílová MAC adresa FF- FF- FF- FF- FF- FF.

RIP zpráva (2.1) se skládá z těchto políček:

- Command – obsahuje 1 nebo 2 podle toho, zda se jedná o žádost (Request) nebo odpověď (Response).
- Version – obsahuje informaci o verzi, v tomto případě jde o RIPv1.
- Address Family Identifier – specifikuje použitou adresní rodinu. IP má identifikátor 2.
- IP Address – obsahuje IP adresu cíle cesty.
- Metric – udává počet skoků do cíle a nabývá hodnot 1 až 16.

Jedna zpráva může nést až 25 záznamů, tedy 25 opakujících se políček Address Family Identifier, IP Adress a Metric.



Obrázek 2.1: Struktura zprávy protokolu RIP.

2.2.8 Nástupce RIPv2

V roce 1994 byla vyvinuta novější verze protokolu RIP, RIP verze 2 (RIPv2). RIPv2 je specifikována v RFC 2453 [8], resp. STD 56. Splňuje základní charakteristiky RIPv1, ale řeší některé jeho nedostatky.

Spolu s cílovou adresou a metrikou se ve zprávách také zasílá maska, tím pádem RIPv2 podporuje VLSM i CIDR. Na rozesílání zpráv už nepoužívá broadcast, ale multicast s IP adresou 224.0.0.9. Díky tomuto vylepšení jsou zprávy zasílány jen směrovačům, které také využívají směrovací protokol RIPv2. Dále umožňuje autentizaci směrovacích informací. RIPv2 také dokáže spolupracovat s jinými směrovacími protokoly, ať už vnitřními (pomocí odkazu na následující skok) nebo vnějšími (prostřednictvím označení cest mimo autonomní systém).

2.3 Protokol OSPF

Dalším protokolem, který máme za úkol simulovat, je Open Shortest Path First (dále jen OSPF). Budeme o něj zjišťovat podobné informace jako o protokolu RIP. Tato kapitola bude mít proto podobnou strukturu jako ta předešlá s výjimkou některých rysů specifických pro tento protokol.e

2.3.1 Stručný popis

OSPF je pravděpodobně nejpoužívanějším protokolem IGP[1] (Interior Gateway Protocol) ve velkých podnikových sítích. OSPF byl vyvíjen organizací IETF od 80. let minulého století. V roce 1991 se dočkal první standardizace, dnes se OSPF běžně používá v jeho verzi 2 specifikovanou v RFC 2338 [9] z roku 1998. Jedná se o protokol typu Link-state.

2.3.2 Typy OSPF zpráv

Dříve, než se podíváme na to, jak OSPF protokol funguje, je dobré udělat si pořádek ve zprávách (resp. paketech), které OSPF používá. Je jich několik druhů – Hello, Database

[1] Směrovací protokol používaný uvnitř autonomního systému (RIP, OSPF, EIGRP atd.).

Description (DBD), Link-State Request (LSR), Link-State Update (LSU) a Link-State Acknowledgement (LSAck).

Hello pakety se používají k nalezení sousedů, ustavení a udržování spojení mezi nimi. DBD slouží k synchronizaci databáze mezi směrovači. LSR zpráva je žádost o určitý záznam Link-state. LSU je pak odpověď na LSR a obsahuje žádaný záznam. LSAck se používá k potvrzení ostatních paketů.

2.3.3 Router ID

Při procesu výměny informací mezi směrovači v OSPF oblasti je nutné jednoznačně identifikovat jednotlivé směrovače. K tomu slouží tzv. Router ID. Hodnota Router ID je automaticky nastavena na nejnižší IP adresu rozhraní Loopback[2]. Pokud směrovač žádné rozhraní Loopback nemá, je použita nejnižší IP adresa ze všech rozhraní. Router ID jde samozřejmě i nastavit ručně v IOS směrovače. Volba Router ID se provede na počátku a už se později nemění.

2.3.4 Funkce protokolu OSPF

Po restartu musí směrovač ze všeho nejdříve objevit své sousední směrovače (Neighbor) a ustanovit s nimi spojení. Rozešle tedy multicastově na všechna svá rozhraní Hello paket. Příjem Hello paketu na některém rozhraní směrovače znamená, že je směrovač připojen k jinému OSPF směrovači na této lince. Předtím, než může dojít k spojení, se musí směrovače shodnout v třech základních bodech – Hello Interval, Dead Interval a Network Types.

Pokud sousední směrovač souhlasí s těmito parametry komunikace, zašle původnímu směrovači odpověď obsahující jeho Router ID. Původní směrovač dostane tuto zprávu a dojde k ustavení spojení. Přestože na začátku se Hello pakety posílají pomocí multicastu, po ustavení spojení spolu směrovače komunikuji pomocí unicastu. Toto ustavení musí proběhnout v obou směrech.

Po ustavení spojení je třeba naplnit topologickou databázi informacemi o topologii sítě. Ve výsledku obsahují databáze všech směrovačů v OSPF síti identické údaje. Rozdíly vznikají až při vytváření směrovacích tabulek. Směrovače si zašlou DBD paket, který obsahuje náhodně vybrané sekvenční číslo, které budou používat k další komunikaci při výměně topologických informací. Používá se sekvenční číslo, které navrhne směrovač s vyšším Router ID.

Následně si posílají další pakety DBR s informacemi ze svých topologických databázích. Směrovač pak porovnává tyto informace ze sousedního směrovače s informacemi, které má ve své vlastní databázi. Pokud zjistí, že mu některá informace v databázi chybí nebo je zastaralá, vyžádá si od svého souseda tyto informace pomocí paketu LSR. Sousední směrovač pošle tyto informace v paketu LSU. Směrovač následně příjem tohoto paketu potvrdí pomocí LSAck. Kdyby k tomuto potvrzení nedošlo, poslal by soused LSU po nějakém času znovu.

Uvedený postup získávání topologických informací se opakuje i v případě, že směrovač zjistí, že některý z jeho sousedu již není aktivní. Vyšle všem svým sousedům LSU paket s informací o změně topologie. Sousedi pak posílají tento LSU dále po síti až se dostane ke všem směrovačům v topologii.

[2]Logická smyčka, umožňující zasílat pakety sám sobě.

2.3.5 Časovače

Protokol OSPF používá několik časovačů, které udržují stav topologické databáze a tím pádem i směrovací tabulky všech směrovačů v konzistentním stavu.

Základními kameny jsou časovače Hello Interval a Dead Interval, na kterých se směrovače musí shodnout před zahájením výměny informací. Hello interval udává, jak často si směrovače zasílají Hello pakety. Implicitně je to 10 sekund v sítích LAN[3] (Local Area Network) a 30 sekund v sítích NBMA[4] (Nonbroadcast Multiple Access Network).

Pokud Hello paket nepřišel do uplynutí Dead Interval, je soused považován za nefunkčního a je třeba o tom informovat zbývající sousedy. Dead Interval je implicitně čtyřnásobek Hello Intervalu, tedy 40 sekund pro LAN sítě a 120 sekund pro NBMA sítě.

Dalším časovačem je MaxAge. Tento časovač je propojen s políčkem Age, který je u každého záznamu v topologické databázi. Ten se periodicky inkrementuje a pokud dosáhne hodnoty MaxAge, je informace v topologické databázi již považována za zastaralou a je odstraněna z databáze. Implicitně je MaxAge nastaven na 1 hodinu.

Aby nedocházelo k odstranění položek z databáze, tak se s periodou časovače LSRefreshTime zasílají sousedům LSA pakety, které vynulují políčko Age u záznamu. Implicitní hodnota tohoto časovače je 30 minut.

2.3.6 Metrika a administrativní vzdálenost

Metrika, která se používá u protokolu OSPF se nazývá Cost nebo-li cena. RFC ale nespecifikuje, co by mělo být použito pro určování této ceny.

Z RFC 2328 [5]: *„A cost is associated with the output side of each router interface. This cost is configurable by the system administrator. The lower the cost, the more likely the interface is to be used to forward data traffic. “*

Vzhledem k tomu, že tato práce se zabývá Cisco technologiemi, je třeba vzít v úvahu cenu, tak jak ji definuje firma Cisco. Ta používá jako cenu součet šířek pásma výstupních rozhraní po cestě od směrovače do cílové sítě. Cena rozhraní se vypočítá jako $10^8/bw$, kde bw je šířka pásma v jednotce bps. Díky tomuto vzorci dosáhneme toho, že rozhraní s větší šířkou pásma budou mít nižší cenu. Jak bylo uvedeno v citaci výše, čím nižší cena, tím je cesta preferovanější.

Z daného vzorce ale vyplývá, že rozhraní FastEthernet (100 Mbps) a rychlejší budou mít stejnou cenu, tj. jedna. Některé implementace proto umožňují konstantu 10^8 zvýšit nebo zadat cenu rozhraní ručně, čímž je možné i priorizovat některé námi vybrané cesty.

Administrativní vzdálenost protokolu OSPF je 110. Tím pádem jsou cesty zjištěné protokolem OSPF považovány za důvěryhodnější a tím pádem i preferovanější než cesty zjištěné protokolem RIP nebo třeba IS-IS.

2.3.7 Vytváření směrovací tabulky

Po naplnění topologické databáze informacemi o topologii sítě je třeba tyto informace transformovat na směrovací tabulku. Tuto transformaci provádí algoritmus Dijkstra's shortest path first (SPF).

Tento algoritmus nejprve vytvoří z databáze graf, přesněji strom SPF. Všechny hrany tohoto stromu ohodnotí cenami (metrikou). Vrcholem stromu je vždy daný směrovač, který

[3] Počítačová síť, která pokrývá malé geografické území, např. domácnost.

[4] Síť propojující několik směrovačů, není však schopna posílat broadcast, např. sítě Frame Relay, ATM nebo X.25.

si tvoří směrovací tabulku. SPF pak vytváří strom takovým způsobem, že se snaží odstranit všechny potenciální smyčky a najít nejlepší možnou cestu do cílové sítě.

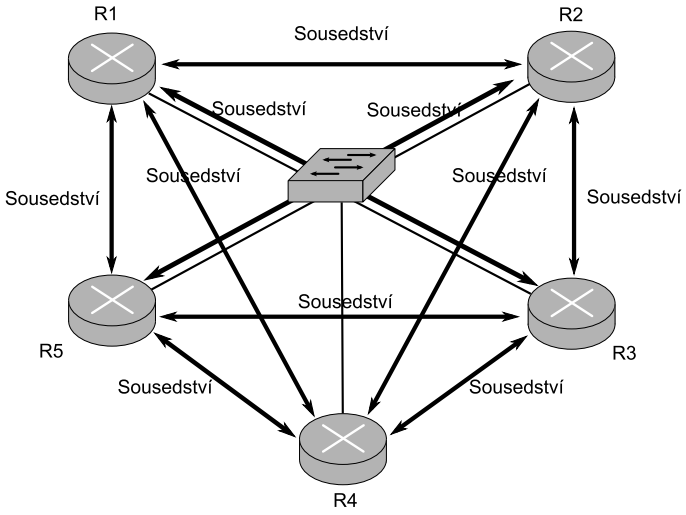
Když už je výsledný strom hotový, do směrovací tabulky se vždy uloží IP adresa cíle, další skok (Next Hop) a celková cena cesty. Pokud SPF algoritmus objeví dvě cesty do cíle, které mají stejnou cenu, vloží do směrovací tabulky obě. Následně záleží na nastavení směrovače, zda si vybere jednu z těchto cest nebo bude využívat obě a část paketů zprávy pošle jednou cestou a druhou část druhou cestou. Druhý uvedený způsob je praktičtější vzhledem k rovnoměrnému zatížení obou linek.

Z tohoto popisu může být jasné, že algoritmus SPF je pro směrovač poměrně náročný a není žádoucí, aby se výpočet opakoval příliš často. To může být problém, pokud se v síti objeví nějaká nestabilní linka, která často padá. Z tohoto důvodu byl stanoven minimální čas mezi výpočty algoritmu SPF.

2.3.8 Designated Router

Z uvedeného popisu funkce OSPF protokolu vyplývá, že pokud mezi směrovači nebudou spoje Point-to-point, ale síť s vícenásobným přístupem (Multiaccess), může být komunikace mezi směrovači náročná a v nejhorším případě může zahltit linku.

Vezměme si jako příklad topologii z obrázku 2.2. Zde je propojeno pět směrovačů. Každý směrovač musí s každým dalším směrovačem sítě ustavit spojení a konfrontovat své záznamy z databáze. Pokud se stane jeden soused nefunkčním, všechny směrovače si navzájem začnou posílat informaci o změně topologie.



Obrázek 2.2: OSPF síť bez Designated Router.

Z obrázku 2.2 je zřejmé, že pro pět směrovačů musí dojít k vytvoření deseti spojením. Z toho vyplývá jednoduchý vztah pro výpočet potřebných spojení, pokud máme v síti s vícenásobným přístupem n směrovačů:

$$\frac{n(n-1)}{2} \tag{2.1}$$

Budeme-li mít v této síti například 100 směrovačů, vyšplhá se celkový počet spojení na 4950! To linku hodně zatěžuje a proto zde existuje řešení v podobě volby Designated Routeru a Backup Designated Routeru.

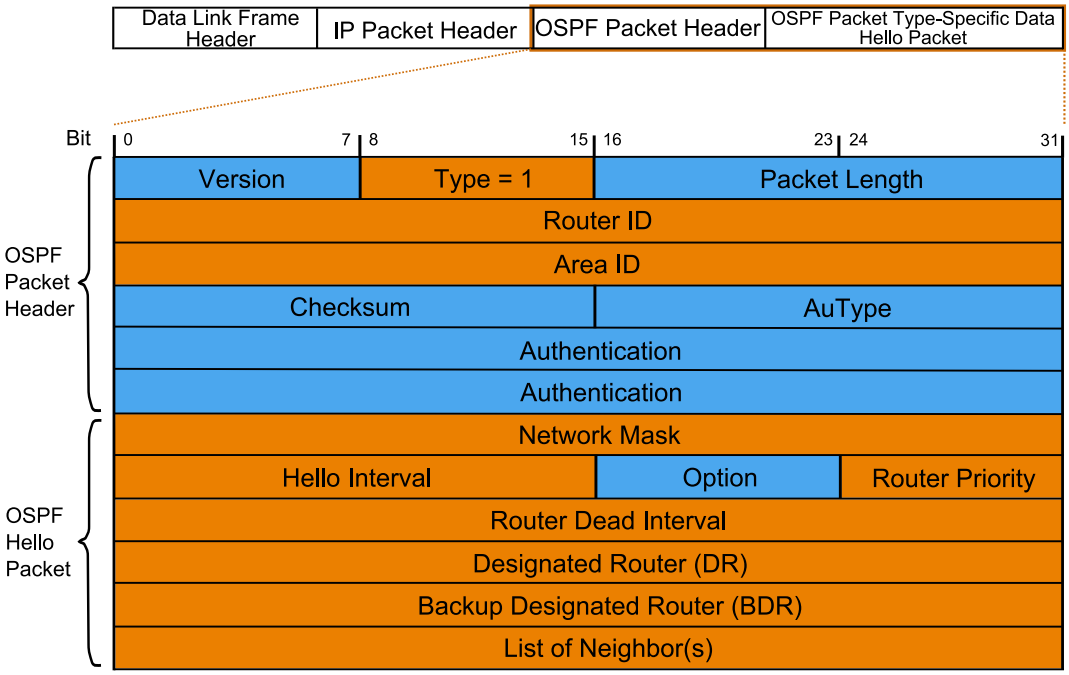
Směrovače si na počátku zvolí Designated Router (DR), což je směrovač s nejvyšší prioritou (Router Priority). Backup Designated Router (BDR), nebo-li záložní DR, je směrovač s druhou nejvyšší prioritou. Informace o prioritě se zasílá na počátku v Hello paketu. Směrovač poté navazuje sousedské vztahy pouze s DR. Následně veškerá komunikace mezi směrovači probíhá tak, že směrovač zašle informaci DR a ten ji rozešle všem ostatním směrovačům v síti. V sítích Point-to-point se pro komunikaci mezi směrovači používá unicast. Zde ale směrovač posílá informace, jak DR, tak i BDR, proto se používá multicastová komunikace.

2.3.9 Formát zprávy

OSPF, narozdíl od protokolu RIP, nepoužívá transportní vrstvu, ale pracuje přímo s protokolem IP. OSPF její funkce nepotřebuje, protože si sám zajišťuje detekci chyb a opravu.

Každá OSPF zpráva je zabalena do IP paketu, jehož hlavička obsahuje cílovou multicastovou adresu 224.0.0.5 nebo 224.0.0.6. V případě, že v síti používáme DR a BDR, tak komunikace s nimi probíhá na IP adrese 224.0.0.6 a komunikace se všemi OSPF směrovači pomocí IP adresy 224.0.0.5. Hlavička Ethernetového rámce pak nese informaci o cílové multicastové MAC adrese 01-00-5E-00-00-05 nebo 01-00-5E-00-00-06.

Z obrázku 2.3 je zřejmé, že OSPF zpráva se skládá z hlavičky a těla nesoucího data. Zde popíši jen jednu z pěti nejčastěji používaných zpráv – Hello paket. Hlavička je však u všech zpráv stejná a skládá se z těchto částí:



Obrázek 2.3: Struktura OSPF Hello paketu.

- Version – identifikuje verzi OSPF, která je použita. Nejčastěji se jedná o verzi 2.
- Type – udává typ OSPF paketu (Hello, DBD, LSR, LSU nebo LSAck).
- Packet length – udává délku paketu včetně hlavičky v bytech.

- Router ID – identifikuje zdroj paketu, jedná se o jednoznačný identifikátor směrovače.
- Area ID – identifikuje oblast (Area), ke které paket patří.
- Checksum – kontrolní součet zajišťuje paket proti chybovosti nebo poničení při přesunu.
- AutType – obsahuje typ autentizace. Všechny OSPF výměny zpráv jsou autentizovány.
- Authentication – nese autentifikační informaci.

Datová část Hello paketu se skládá z těchto částí:

- Network Mask – síťovou masku související s rozhraním, skrz které je zpráva zasílána.
- Hello Interval – udává interval mezi zasíláním Hello paketů v sekundách.
- Router Dead Interval – udává čas v sekundách, po jejichž uplynutí, pokud nepřijde Hello paket, je soused odstraněn z topologické databáze.
- Router Priority – používá se při volbě DR a BDR.
- Designated Router (DR) – obsahuje identifikaci (Router ID) DR směrovače.
- Backup Designated Router (BDR) – obsahuje identifikaci (Router ID) BDR směrovače.
- List of Neighbor(s) – nese seznam identifikací (Router ID) sousedních směrovačů.

2.4 Návrhové vzory Cisco

Aby síť, kterou budeme vzorově simulovat, byla co nejvěrnější a odpovídala využití v praxi, použijeme pro vytvoření topologií návrhové vzory společnosti Cisco. V této části se proto budeme zabývat hledáním a výběrem vhodných návrhových vzorů pro simulaci. Návrhový vzor je vzorová topologie, kterou Cisco doporučuje využít při návrhu sítí. Na konci se krátce zmíníme o principu a smyslu redistribuce.

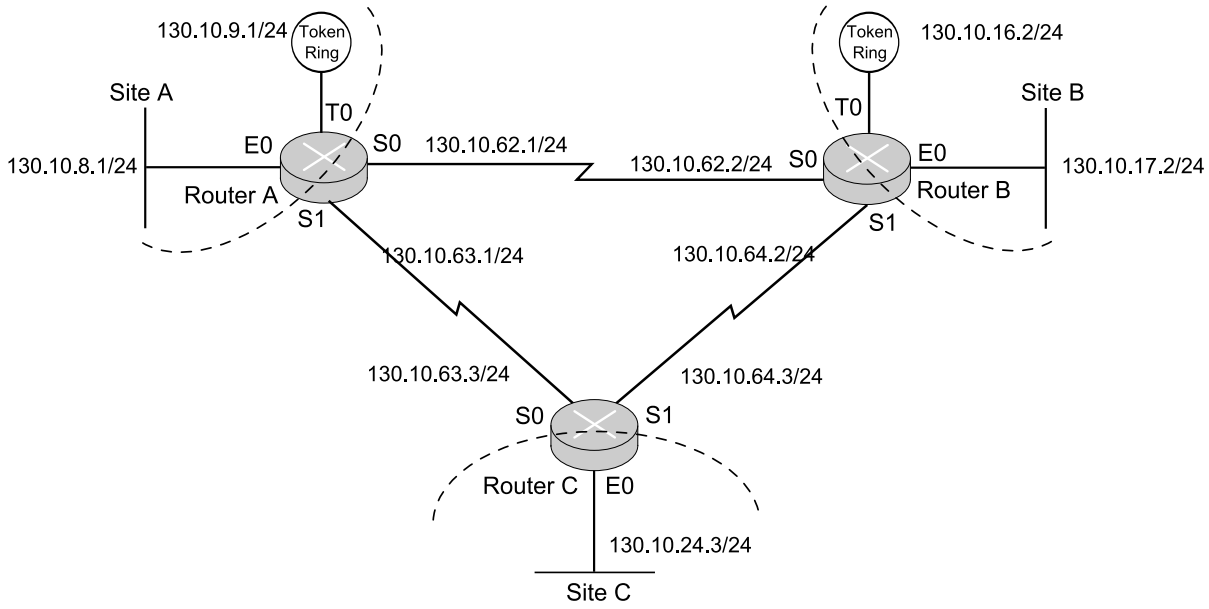
2.4.1 Výběr vhodných návrhových vzorů

Cílem této práce je prostudovat návrhové vzory od firmy Cisco se zaměřením na RIP a OSPF. Pro ideální propojení těchto dvou protokolů jsem se při hledání zaměřila na návrhové vzory zabývající se redistribucí mezi těmito protokoly. Na webových stránkách firmy Cisco lze nalézt mnoho návrhových vzorů.

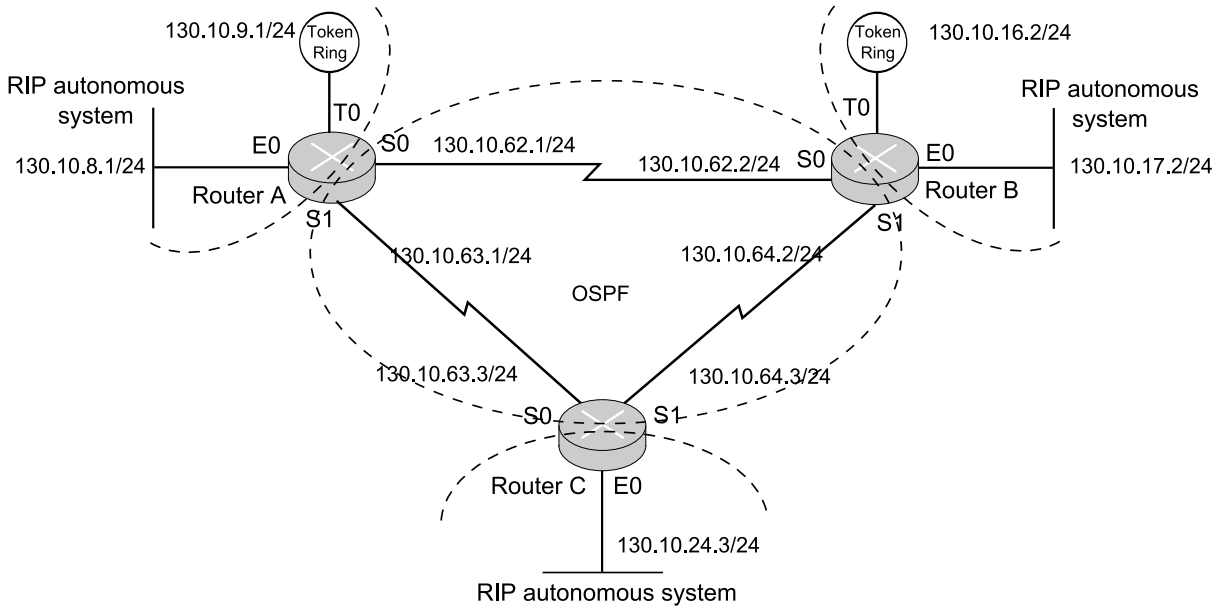
Mým požadavkům vyhovoval [1]. V tomto dokumentu jsou tři návrhové vzory, které budu modelovat a simulovat. Všechny topologie se skládají z páteře a okolí. Páteř se skládá ze tří směrovačů propojených mezi sebou. Sítě LAN, ve kterých se předpokládají uživatelé a jiná koncová zařízení, se napojují na tyto páteřní směrovače.

V prvním návrhovém vzoru (obrázek 2.4) je v síti použit pouze směrovací protokol RIP. RIP je třídní protokol, proto jsou zde použity IP adresy třídy B.

Druhý návrhový vzor (obrázek 2.5) je dalším krokem v přechod z RIP protokolu na OSPF protokol. Páteř bude provozována na protokolu OSPF a LAN sítě na protokolu RIP. Tedy tři páteřní směrovače musí být hraničními směrovači a je požadováno, aby uměly pracovat s oběma směrovacími protokoly a tak mohly mezi nimi provádět redistribuci.

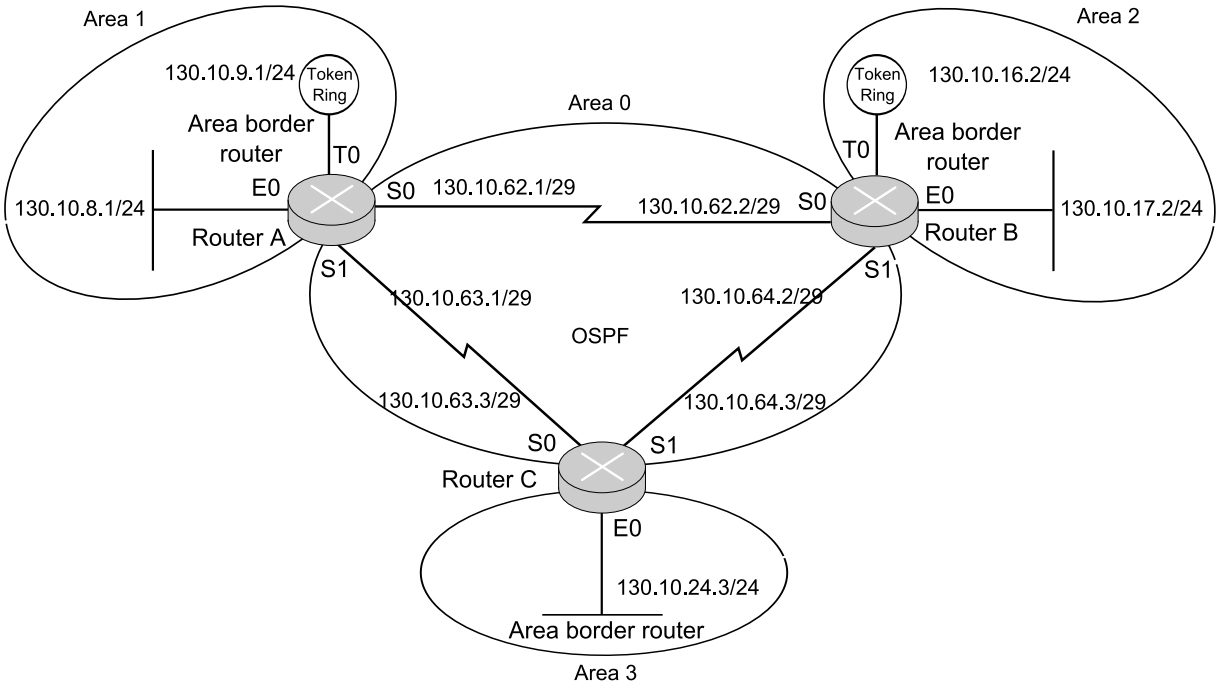


Obrázek 2.4: Síť se směrovacím protokolem RIP.



Obrázek 2.5: RIP síť se směrovacím protokolem OSPF ve středu.

Třetí návrhový vzor (obrázek 2.6) představuje síť používající již jen směrovací protokol OSPF. Návrh ale využívá několik autonomních oblastí. Autonomní oblast páteřní části je `area0` a každá ze sítí připojených k páteřním směrovačům tvoří další autonomní oblastí `area1` – `area3`. Páteřní směrovače jsou oblastními hraničními směrovači, které kontrolují výměnu směrovacích informací mezi oblastmi.



Obrázek 2.6: Síť rozdělená na OSPF oblasti.

2.5 Shrnutí kapitoly

Tato kapitola shrnula základní poznatky o směrování, směrovacích protokolech a redistribuci. Dále jsme si popsali směrovací protokoly RIP i OSPF, které jsou velmi populární a často užívané, avšak každý z nich je vhodný pro jiné účely.

Protokol RIP je nejstarším a nejoblíbenějším směrovacím protokolem. Hlavní jeho výhodou je jednoduchost a velmi snadná implementace, které často vyváží jeho nedostatky. Hlavní nedostatek je absence masek, díky které RIPv1 nepodporuje VLSM a CIDR. Tyto nevýhody jsou zcela odstraněny v novější verzi RIPv2. Avšak zásadní nevýhodou stále zůstává metrika, která nabývá 1 až 15 skoků a tedy není možné nasadit protokol RIP v rozsáhlých sítích. Oblíbenost protokolu je zřejmá, neboť se stále vyvíjí a již existuje i verze pro IPv6 (RIPng).

OSPF je nejpoužívanějším protokol typu Link-state. Oproti protokolu RIP se může pochlubit rychlejší konvergencí, možností zasílat masky sítě (tím pádem i podpora CIDR a VLSM) a obecně možností použití ve větších sítích. Po ustavení spojení komunikují směrovače pomocí unicastu (Point-to-point sítě) nebo multicastu (Multiaccess sítě), což je výhodné pro malé zatížení linky. OSPF také podporuje autentizaci s šifrováním hesla MD5, která zajišťuje ověřování pravosti předávaných informací. Nemůže tedy dojít k podvržení informace a následnému přesměrování provozu. Hlavní nevýhodou je náročnost algoritmu SPF a tím pádem neschopnost reagovat na příliš časté změny a také vysoká náročnost kladená na směrovač při výpočtu. Proto jsou zařízení podporující OSPF dražší. Větší firmy a instituce si ale jistě rádi připlatí a získají tak výhody tohoto protokolu. Perspektiva protokolu je zřejmá, neboť se stále vyvíjí a již existuje i verze pro IPv6 (OSPFv3). K dokreslení přikládám srovnávací tabulku 2.1.

Na stránkách firmy Cisco jsem vybrala návrhové vzory týkajících se užití protokolů RIP

a OSPF ve větších (firemních, školních) sítích. Použití obou protokolů v jedné síti je pro jejich simulaci ideální a proto jsem se zaměřila i na jejich vzájemnou redistribuci. Popsané topologie jsou vhodné pro simulaci.

Vlastnosti	**RIPv1**	**RIPv2**	**OSPF**
Typ protokolu	Distance Vector	Distance Vector	Link-state
Konvergence	pomalá	pomalá	rychlá
Metrika	počet skoků	počet skoků	cena cesty
VLSM	ne	ano	ano
CIDR	ne	ano	ano
Zatížení směrovače	malé	malé	velké
Zatížení sítě	velké	velké	malé

Tabulka 2.1: Srovnání směrovacích protokolů RIP a OSPF

Kapitola 3

Simulační modely

Co je to modelování a simulační model jsme si již vysvětlili v sekci 1.1. Nyní můžeme převést teorii do praxe a za použití nástrojů PT (sekce 1.2) a OMNeT++ (sekce 1.3) vytvořit simulační modely topologií sítí z vybraných návrhových vzorů (sekce 2.4).

3.1 Simulační modely v PT

V této sekci si vysvětlím, jak jsme vytvářeli dle návrhů simulační modely v programu PT.

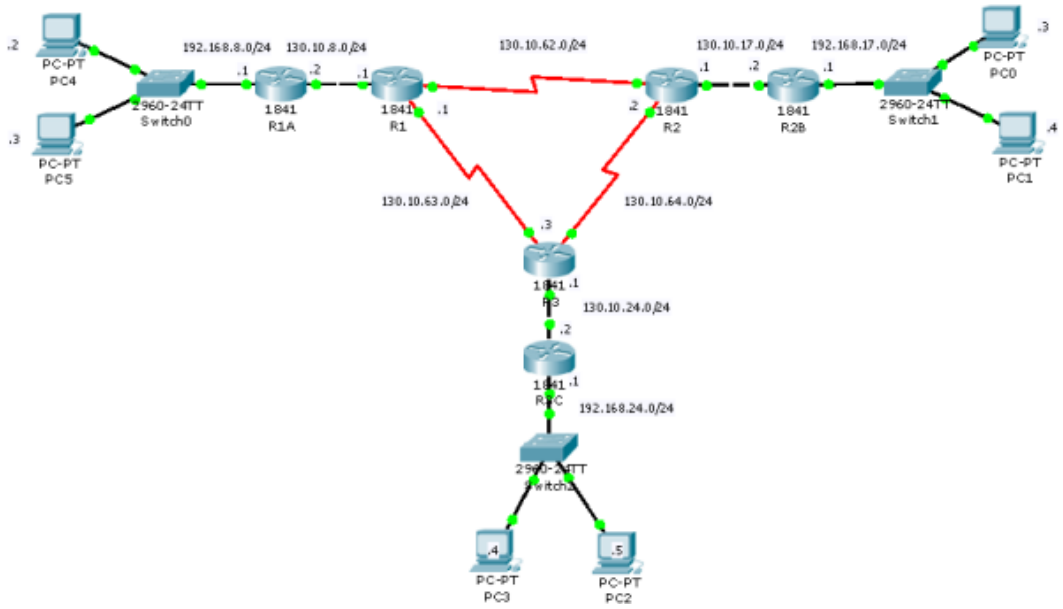
3.1.1 Tvorba modelů

Pro vytvoření modelu jsem použila Cisco směrovače 1841 s modulem WIC-2T a přepínače 2960. Směrovače jsem mezi sebou propojila sériovým kabelem a k směrovačům jsem připojila kříženým kabelem další směrovače představující podsítě a k nim přímým kabelem po jednom přepínači. Modelově jsem ke každému přepínači připojila přímým kabelem po dvou počítačích. Na obrázku 3.1 je vidět výsledná topologie.

Následovalo nakonfigurování sítě. Ke konfiguraci směrovačů jsme použili IOS. Výpis konfiguračního registru pro první model (pouze RIP) na směrovači R1 je následující:

```
interface FastEthernet0/0                ;konfigurace ethernetového portu
ip address 130.10.8.1 255.255.255.0      ; nastavení IP adresy a masky
duplex auto
speed auto
!
interface Serial0/1/0                    ; konfigurace sériového portu
ip address 130.10.62.1 255.255.255.0     ; nastavení IP adresy a masky
clock rate 56000                         ; kmitočet (u DCE)
!
interface Serial0/1/1
ip address 130.10.63.2 255.255.255.0
clock rate 56000
!
router rip                               ; konfigurace protokolu RIP
network 130.10.0.0                       ; distribuovaná síť
```

Nastavení na směrovači R1 pro druhý model (RIP + OSPF) je:



Obrázek 3.1: Topologie sítě v PT.

```
interface FastEthernet0/0
ip address 130.10.8.1 255.255.255.0
duplex auto
speed auto
!
interface Serial0/1/0
ip address 130.10.62.1 255.255.255.0
clock rate 56000
!
interface Serial0/1/1
ip address 130.10.63.2 255.255.255.0
clock rate 56000
!
router ospf 100                              ; konfigurace protokolu OSPF
log-adjacency-changes
redistribute rip subnets                     ; redistribuce do RIP
network 130.10.62.0 0.0.0.255 area 0         ; distribuované sítě
network 130.10.63.0 0.0.0.255 area 0
!
router rip                                ; konfigurace protokolu RIP
redistribute ospf 100  metric 4           ; redistribuce do OSPF s metrikou 4
passive-interface Serial0/1/0             ; rozhraní, na které se nebudou ...
passive-interface Serial0/1/1             ; ... zasilat RIP zprávy
network 130.10.0.0                        ; distribuovaná síť
```

Nastavení na směrovači R1 v třetím modelu (OSPF oblasti) je:

```
interface FastEthernet0/0
ip address 130.10.8.1 255.255.255.0
duplex auto
speed auto
!
interface Serial0/1/0
ip address 130.10.62.1 255.255.255.248
clock rate 56000
!
interface Serial0/1/1
ip address 130.10.63.1 255.255.255.248
clock rate 56000
!
router ospf 100
log-adjacency-changes
redistribute rip subnets
network 130.10.62.0 0.0.0.255 area 0
network 130.10.63.0 0.0.0.255 area 0
network 130.10.8.0 0.0.0.255 area 1
```

3.2 Simulační modely v OMNeT++

V této sekci vysvětlíme tvorbu simulačních modelů v nástroji OMNeT++. OMNeT++ jako takový nemá téměř žádné možnosti pro směrování a nepodporuje protokoly RIP a OSPF. Proto použijeme framework INET, který má podporu OSPF a obsahuje další důležité implementace pro síťový provoz. Protokol RIP zcela chybí a bude ho třeba doimplementovat.

3.2.1 Modelování návrhových vzorů

Nejprve si vytvoříme složky pro budoucí simulační modely. Pro každý návrhový vzor vytvoříme model sítě v jazyce NED. Jedná se o složený modul. Popis všech tří modelů je podobný.

Použijeme složený modul směrovače `ANSARouter`, který vznikl v rámci bakalářské práce V. Siváka [14]. Kromě tohoto modulu využijeme jednoduché moduly síťových prvků z knihovny INET a to koncové zařízení typu `StandardHost` a přepínače typu `EtherSwitch`. Nakonec provedeme propojení jejich bran, resp. síťových rozhraní. Část `RIP.ned` souboru popisující topologii prvního modelu (viz. obr. 2.4) je:

```
import inet.nodes.ethernet.EtherSwitch;      ; vložené knihovny
import inet.ansa.ANSARouter;
import inet.nodes.inet.StandardHost;
import inet.world.ChannelInstaller;
import inet.world.ScenarioManager;
import ned.DatarateChannel;

network RIP                                  ; název modelované sítě
{
    <část kódu vypuštěna>
```

```
    submodules:                                 ; moduly, ze kterých se skládá
        <část kódu vypuštěna>
        H11: StandardHost {                     ; PC
            parameters:                         ; obrázek a jeho umístění
                @display("p=40,52;i=device/pc2");
            gates:
                ethg[1];                        ; jedno ethernetové rozhraní
        }
        S1: EtherSwitch {                       ; přepínač
            parameters:
                @display("p=168,92;i=device/switch");
            gates:
                ethg[3];                        ; má tři ethernetové rozhraní
        }
        R1: ANSARouter {                        ; směrovač
            parameters:
                @display("p=357,93;i=srouter");
            gates:
                ethg[3];                        ; má tři ethernetové rozhraní
        }
        <část kódu vypuštěna>
    connections:                                ; propojení modulů skrz brány
        H11.ethg[0] <--> C <--> S1.ethg[1];
        H21.ethg[0] <--> C <--> S2.ethg[1];
        S1.ethg[0] <--> C <--> R1A.ethg[0];
        S2.ethg[0] <--> C <--> R2B.ethg[0];
        R1.ethg[0] <--> C <--> R1A.ethg[1];
        R2.ethg[0] <--> C <--> R2B.ethg[1];
        <část kódu vypuštěna>
}
```

Pro každý model je dále nutné vytvořit soubor `omnetpp.ini`, který obsahuje parametry pro spuštění simulace. Nakonec pro modely vytvoříme spustitelný soubor `run`. Do tohoto souboru stačí napsat:

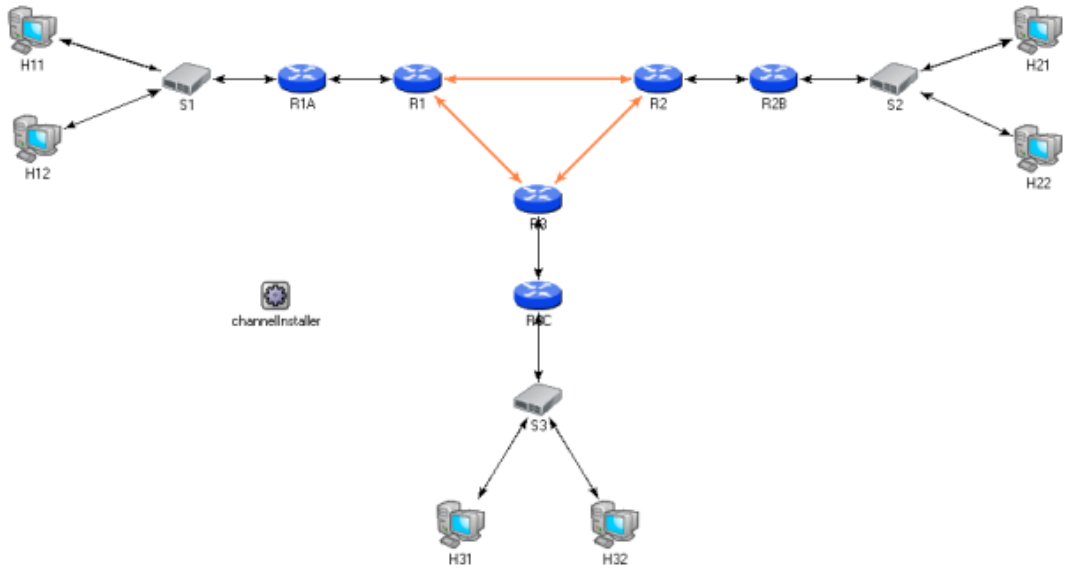
```
#!/bin/sh
../../../../src/run_inet $*
```

Soubor se odkazuje na přeloženou knihovnu INET. Spuštění souboru zajistí načtení nastavení ze souboru `omnetpp.ini` a spuštění simulačního prostředí.

3.2.2 Příprava modelů pro simulaci

Plně funkční vytvoříme zatím pouze třetí model (viz. obr. 2.6), který využívá směrovací protokol OSPF a směrování mezi čtyřmi oblastmi (Area 0 - 3). Přestože INET obsahuje implementaci protokolu OSPF, použijeme upravenou verzi OSPF, která vznikla v rámci bakalářské práce [4].

V předchozí sekci jsme si vytvořili soubor popisující topologii sítě `ospfAreas.ned`. Výslednou topologii je možné vidět na obrázku 3.2.



Obrázek 3.2: Topologie sítě v OMNeT++.

Pro konfiguraci sítě využíváme XML soubor RoutersConfig, jehož struktura byla navržena v rámci bakalářské práce P. Scherfela [13]. V něm přiřadíme jednotlivým rozhraním směrovačů IP adresy a masky a provedeme jejich rozdělené do oblastí OSPF.

Dále vytvoříme pro zařízení pracující se směrovací tabulkou (PC, směrovač) soubory obsahující konfiguraci síťových rozhraní a statické cesty. Tyto informace se nacházejí v souborech `.irt`. Název souboru je pro každé zařízení uloženo v inicializačním souboru `omnetpp.ini` v parametru `routingFileName`. U každém síťového rozhraní napíšeme jméno rozhraní, IP adresu, masku, multicastové skupiny, MTU a metriku.

Poté, co naimplementujeme protokol RIP a redistribuci, doplníme zbývající dva modely pro OMNeT++ podobným způsobem, jaký byl v této sekci uveden. Konfigurační soubory k jednotlivým modelům jsou uvedeny v příloze D.

3.3 Shrnutí kapitoly

V této kapitole jsme vytvořili modely v programu PT. Vyvářené modely jsme nakonfigurovali v IOS stejně jako v na reálném zařízení, což je velká výhoda programu PT. Vytváření modelů bylo velmi lehké a rychlé. Vznikli nám tak tři soubory `.pkt`, které jsou připraveny pro simulaci.

V OMNeT++ se simulační modely vytvářely hůř a to především kvůli nutnosti popsat modely v málo známém jazyku NED. Plně funkční je pouze model využívající protokol OSPF. Pro ostatní návrhové vzory bude třeba doimplementovat části umožňující směrovaní pomocí protokolu RIP a umožňující redistribuci mezi RIP a OSPF.

Kapitola 4

Implementace

Tato kapitola se zabývá implementací jednoduchých modulů do framework INET pro směrovací protokol RIP a pro redistribuci směrovacích informací mezi protokoly RIP a OSPF.

4.1 Implementace protokolu RIP

V této sekci bude popsána postupná implementace směrovacího protokolu RIP v jeho první verzi. Bude se zabývat popisem jednotlivých tříd, ze kterých se jeho implementace skládá. Při implementaci jsme vycházeli z RFC 1058 [5].

4.1.1 RIPPacket

Pro správnou funkčnost směrovacího protokolu si nejprve vytvoříme zprávu, která ponese všechny informace, které nese i reálná zpráva RIP. Zpráva `RIPPacket.msg` plně koresponduje se standardem (obrázek 2.1). Po překladu se automaticky vygeneruje příslušný zdrojový (`.cc`) a hlavičkový soubor. `RIPPacket.msg` vypadá následovně:

```
struct RouteEntry                   ; záznamy o cestách
{
    short      addressID;           ; typ adresy (2 pro IP)
    short      mustBeZero2 = 0;
    IPAddress  ipAdress;            ; adresa
    long       mustBeZero3 = 0;
    long       mustBeZero4 = 0;
    long       metric;              ; metrika
}

message RIPPacket
{
    char       command enum(RIPCommand);     ; typ zprávy (Req/Resp)
    char       version = 1;                  ; verze protokolu RIP
    short      mustBeZero1 = 0;
    RouteEntry routeEntry[];                 ; záznamy o cestách
}
```

4.1.2 RIPTimer

Jak jsme si uvedli v podsekci 4.1.2, protokol RIP využívá tři časovače. V diskrétní simulaci se časovače modelují jako zprávy, které objekt zasílá sám sobě se zpožděním. Proto jsme pro ně vytvořili zprávu `RIPTimer.msg`. Struktura zprávy je:

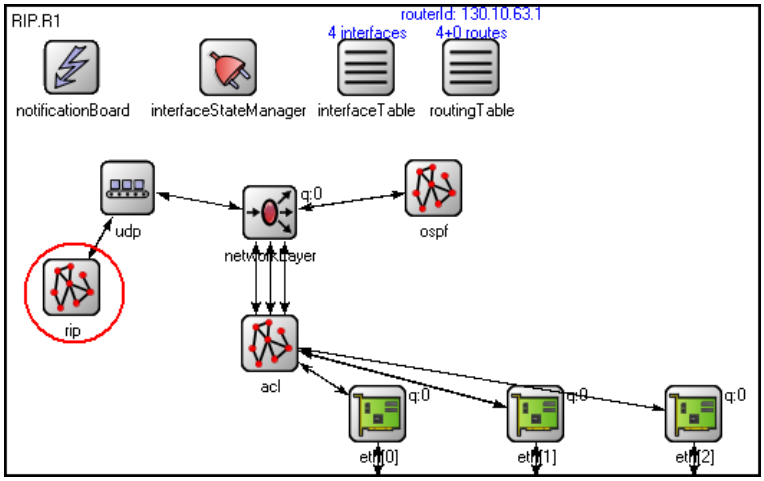
```
enum RIPTimerType       ; typ časovače
{
   hello = 1;           ; časovač Update
   timeout = 2;         ; časovač Timeout
   garbage = 3;         ; časovač Garbage
   trigger = 4;         ; odpočet před rozesláním změn
};

message RIPTimer extends cMessage
{                                                    ; typ časovače
   char  timerKind enum(RIPTimerType) = hello;
};
```

Význam časovačů Update, Timeout a Garbage byl popsán v podsekci . Časovač Trigger se využívá pro rozesílání aktualizací při významné změně (pád/obnovení linky, změna cesty).

4.1.3 RIPRouting

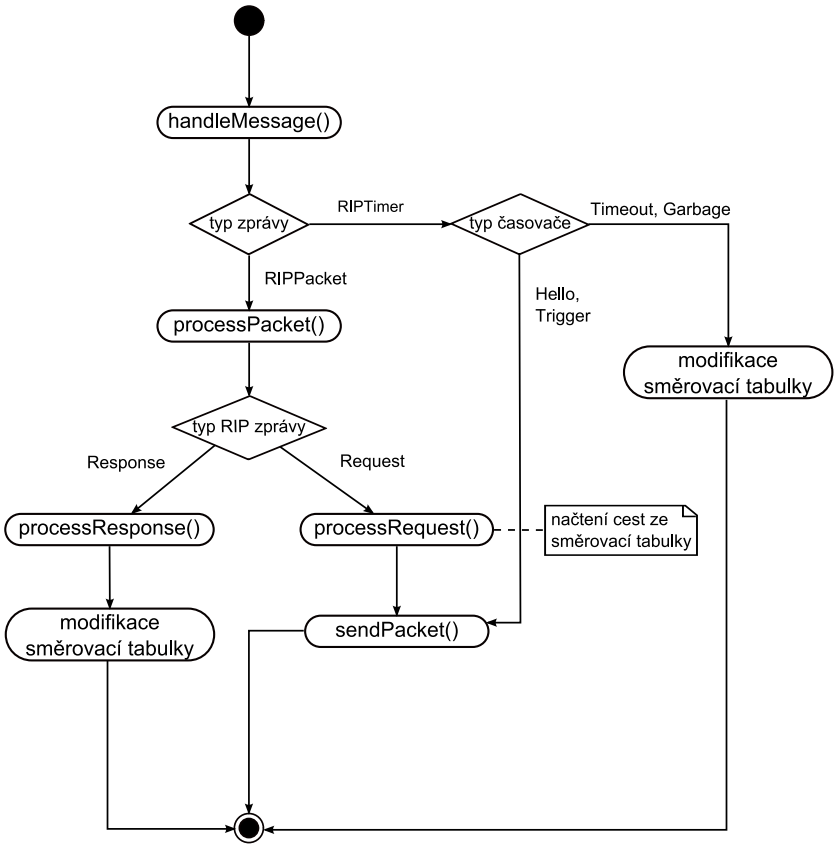
Pro jednoduchý modul RIPRouting jsme vytvořili popis v souboru `RIPRouting.ned` a provedli jeho implementaci v jazyce C++. Jedná se o třídu, která je odvozena od třídy `UDPAppBase`. Ta nám poskytuje prostředky transportního protokolu UDP, který RIP využívá. Jak můžeme vidět na obrázku 4.1, RIP modul je připojen na UDP modul. Hlavičkový soubor třídy je uveden v příloze C.



Obrázek 4.1: Modul ANSARouter obsahující modul RIP.

Metoda `handleMessage()` příjme zprávu a rozpozná, zda jde o zprávu odeslanou z tohoto (některý z časovačů RIP) nebo jiného procesu. V případě, že se jedná o časovač, provede požadované akce (odeslání zprávy, smazání cesty ze směrovací tabulky, restartování časovače).

V případě, že se jedná o cizí zprávu, přepošle ji metodě `processPacket()`. Ta zkontroluje IP adresu, verzi protokolu, strukturu zprávy a zjistí, zda se jedná o žádost (Request) nebo odpověď (Response). Podle toho zprávu dále zpracovává metoda `processRequest()` nebo `processResponse()`.



Obrázek 4.2: Diagram aktivit zobrazující vztahy mezi metodami ve třídě RIPRouting.

Metoda `processRequest()` zkontroluje počet záznamu v žádosti. Pokud je jen jedna, žádá odesílatel o všechny záznamy z směrovací tabulky. Zavolá metodu `sendPacket()`, která se postará o sestavní a zaslání výsledné zprávy sousedovi. Tato metoda využívá metodu `createPacket()` pro vytvoření zprávy a naplnění zprávy směrovacími informacemi.

Metoda `processResponse` zpracovává zprávy se směrovacími informacemi, které přicházejí od sousedních směrovačů. Nejprve zkontroluje, jestli je každý záznam legální, tedy jestli se nejedná o IP adresu smyčky (Loopback), adresy třídy E nebo D apod. Následně dojde k porovnání adresy s adresami v směrovací tabulce. Pokud se v ní taková adresa ještě nevyskytuje, vytvoří se nový záznam a nastaví se mu časovač Timout. Pokud již adresa sítě v tabulce existuje, zkontroluje se metrika. V případě, že záznam má menší metriku, než záznam v tabulce, dojde k aktualizaci záznamu. Nakonec dojde k restartování časovače Timeout.

Celý postup zobrazuje diagram aktivit na obrázku 4.2. Příklad konfigurace je uveden v příloze D.

4.2 Implementace redistribuce

V rámci této práce byla naprogramována redistribuce cest z protokolu OSPF do protokolu RIP. Tuto činnost provádí metoda `getOSPFRoutes()`. Pro její správnou činnost byla vytvořena struktura `RIPRedistribution`:

```
struct RIPRedistribution
{
   bool redistrinute;       ; je nastavena redistribuce?
   char * protocol;         ; redistribuovaný protokol
   int  metric;             ; metrika redistribuovaných cest
};
```

Při inicializaci se do struktury načtou z konfiguračního souboru informace o redistribuci. Zaznamenává, jestli vůbec bude redistribuce probíhat, s jakým protokolem bude redistribuce probíhat (v našem případě pouze OSPF), a jakou metriku budou mít nové RIP záznamy po redistribuci.

Metoda `getOSPFRoutes()` zkontroluje, jestli položka `protocol` je nastavena na `ospf`. Pokud ano začne procházet záznamy směrovací tabulky a kontrolovat jejich zdroj na hodnotu `OSPF`. Takové cesty zaznamená do struktury `RouteEntry` zmíněné u zpráv `RIPPacket` a metriku doplní na hodnotu uloženou ve struktuře `RIPRedistribution`. Takto vytvořené záznamy se pak předávají k dalšímu zpracování, začlenění do RIP zprávy a odeslání sousedům. Příklad konfigurace redistribuce pro jeden směrovač je:

```
<Rip>
   <Redistribute>
      <Protocol>ospf</Protocol>  ; redistribuovaný protokol
      <Metric>4</Metric>         ; RIP metrika redistribuovaných cest
   </Redistribute>
</Rip>
```

4.3 Shrnutí kapitoly

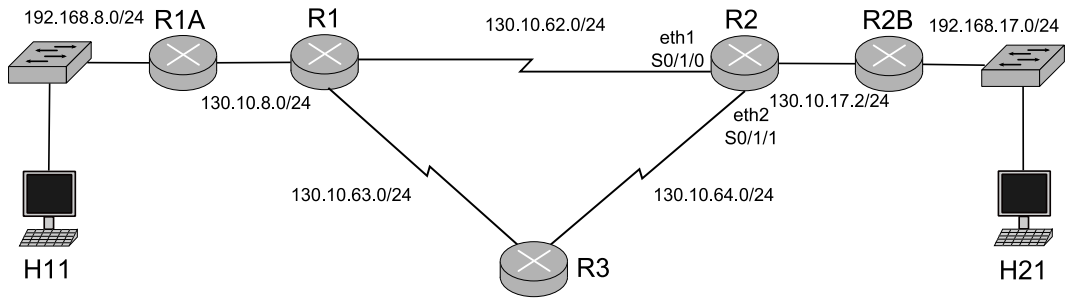
Tato kapitola nastínila implementaci protokolu RIP a redistribuce. Protokol RIP byl implementován dle RFC 1058 [5]. Implementace splnila všechny jeho požadavky až na autosumarizaci. Ta by případně mohla být v budoucnu doimplementována. Pro naše potřeby simulace bychom ji ale nevyužili a aktuální implementace nám bude plně dostačovat. Dále byl naimplementován Split Horizon. V budoucnu by bylo možné implementaci rozšířit také o další ochranný prostředek Poison Reverse.

V rámci této práce byla naimplementována redistribuce z protokolu RIP do protokolu OSPF. Pro plné využití redistribuce se počítá s využitím implementace redistribuce z protokolu RIP do protokolu OSPF, která je součástí jiné bakalářské práce [4], a implementace z bakalářské práce [14], která zajistí zasílání informací o změně v směrovací tabulce.

Kapitola 5

Simulace v prostředí Packet Tracer a OMNeT++

V této kapitole prakticky využijeme naimplementované třídy a simulační modely. Nejprve si definujeme vlastnosti, které chceme ověřit simulací: dostupnost a stabilita sítě. Následně provedeme simulaci v obou zmiňovaných nástrojích – Packet Tracer a OMNeT++. Pro popis simulaci těchto vlastností použijeme referenční obrázek 5.1.



Obrázek 5.1: Topologie simulovaní sítě.

5.1 Dostupnost

Základním požadavkem pro každou síť je dostupnost. Abychom mohli síť plně využívat, musíme si být jisti, že za jakýchkoliv podmínek jsou všechny segmenty sítě odkudkoliv dostupné. Proto je vhodné využívat záložní linky a podobná opatření. V následujících simulacích budeme uvažovat, že dojde k pádu linky mezi směrovači R1 a R2. Dostupnost mezi sítěmi 192.168.8.0/24 a 192.168.17.0/24 by měl záložně zajistit směrovač R3. Ověřovat ji budeme zasíláním ICMP[1] paketů mezi oběma sítěmi.

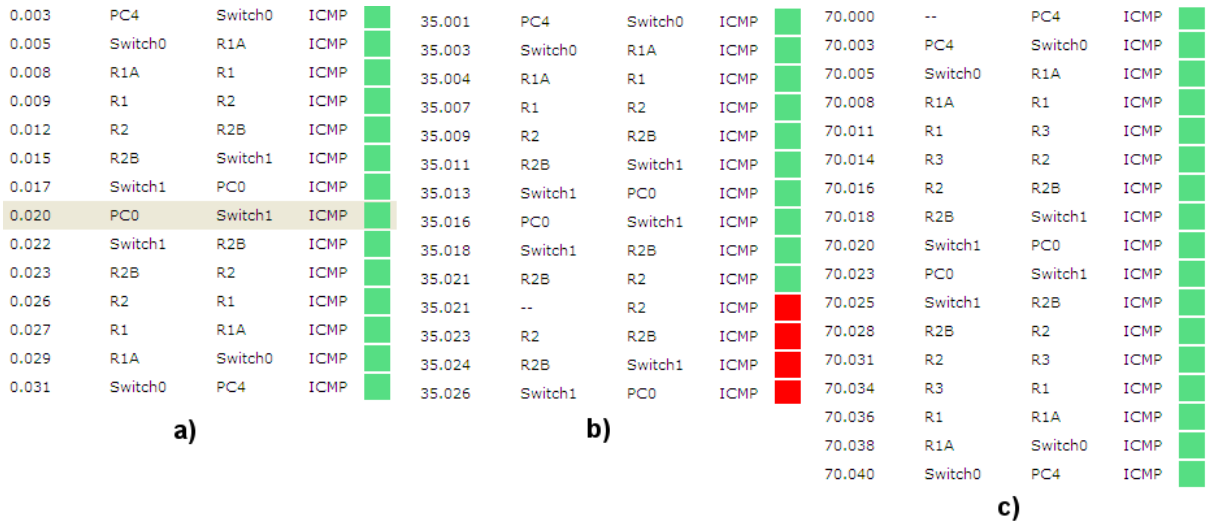
5.1.1 Simulace dostupnosti v nástroji Packet Tracer

V PT jsme si vytvořili pomocí funkce Add Complex PDU scénář, ve kterém bude počítač PC4 periodicky (35 sekund) zasílat ICMP pakety počítači PC0, viz. obrázek 5.1. Spustili

[1] Anglicky Internet Control Message Protocol, používá se pro odesílání chybových zpráv (požadovaná služba/host/síť není dostupná apod.).

jsme simulaci a sledovali průběh cesty paketu. Podle očekávání byl paket směrován přes směrovače R1 a R2 (obrázek 5.2a). V simulačním čase 35.011 jsme zastavili simulaci, přešli jsme do CLI[2] směrovače R2 a pomocí příkazů

```
R2(config)#int s0/1/0
R2(config-if)#shutdown
```



Obrázek 5.2: Výpis průchodu ICMP paketu síťovými zařízeními. a) Linky jsou v pořádku. b) Při cestě paketu zpět došlo k pádu linky. c) Linka mezi R1 a R2 je nefunkční.

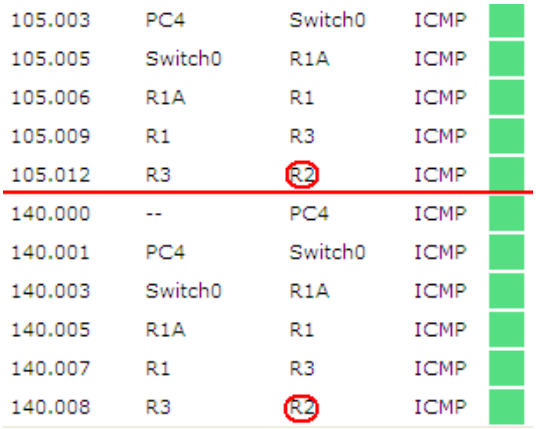
jsme vypnuli rozhraní s0/1/0 vedoucí k směrovači R1. ICMP paket se v té době nacházel v počítači PC0. Při cestě zpět k PC4 směrovač R2 ještě neznal novou cestu do sítě 192.168.8.0 /24, proto směrovač vrátil počítači PC0 ICMP paket typu Destination Unreachable (obrázek 5.2b). Další ICMP paket, který byl vyslán v čase 70 sekund, byl směrován novou cestou, která se mezitím ustanovila, tedy přes směrovače R1, R3 a R2 (obrázek 5.2, c).

Simulaci jsme znovu zopakovali. Tentokrát jsme umístili na rozhraní s0/1/1 (ve směru od R3) směrovače R2 ACL. Zadali jsme ho v CLI takto:

```
R2(config)#access-list 101 deny icmp any 192.168.17.0 0.0.0.255
R2(config)#access-list 101 permit ip any any
R2(config)#
R2(config)#int s0/1/1
R2(config-if)#ip access-group 101 in
```

Po té co byla simulace spuštěna se všemi funkčními linkami, se síť chovala jako při předchozím testu. Po té jsme znovu vypnuli rozhraní mezi směrovači R1 a R2. Tentokrát se síť 192.168.17.0/24 stala nedostupnou (obrázek 5.3). Přestože paket byl předán ze směrovače R1 na R3, tak následující směrovač R2 dle nastaveného ACL paket zahodil.

[2]Rozhraní s příkazovou řádkou určené pro konfiguraci Cisco zařízení, zejména směrovačů a přepínačů.



Obrázek 5.3: Výpis průchodu ICMP paketu síťovými zařízeními. ACL na směrovači R2 paket zahodil.

5.1.2 Simulace dostupnosti v nástroji OMNeT++

Pro simulace v OMNeT++ využijeme manažer stavů linek (třída `InterfaceStateManager`), který vznikl v rámci bakalářské práce M. Danka [4]. Tato třída umožňuje vytvoření scénáře simulace pomocí XML souboru. Zde je ukázka XML souboru, který využijeme pro tuto simulaci:

```
<?xml version="1.0"?>
<scenario>
   <at t="200">
      <interfacedown module="R2.interfaceStateManager" int="eth1"/>
   </at>
   <at t="300">
    <interfaceup module="R2.interfaceStateManager" int="eth1"/>
   </at>
</scenario>
```

V souboru uvádíme, že v simulačním čase 200 sekund dojde k vypnutí rozhraní `eth1` na směrovači R2. V simulačním čase 300 sekund dojde k jeho opětovnému zapnutí.

Pro zasílání ICMP paketů jsme využili aplikaci Ping (třída `pingApp`) na počítači H11. Ten v pravidelných intervalech zasílá ICMP pakety typu Echo počítači H21 (obrázek 5.1). Simulace vypisovala trasy, kterými putoval ICMP paket:

```
-----------------------------------------------
Time = 19.208472580648
H11 : Ping 0 to 192.168.17.21
Path: H11 - DROP
-----------------------------------------------
<výpis vypuštěn>
-----------------------------------------------
Time = 89.208472580648
```

```
H11 : Ping 2 to 192.168.17.21
Path: H11 - R1A - R1 - R2 - R2B - H21 - H21 - R2B - R2 - R1 - R1A - H11
-------------------------------------------------
<výpis vypuštěn>
-------------------------------------------------
Time = 194.208472580648
H11 : Ping 5 to 192.168.17.21
Path: H11 - R1A - R1 - R2 - R2B - H21 - H21 - R2B - R2 - R2 - R2B - H21 -
- ICMP ERROR
-------------------------------------------------
Time = 229.208472580648
H11 : Ping 6 to 192.168.17.21
Path: H11 - R1A - R1 - R3 - R2 - R2B - H21 - H21 - R2B - R2 - R3 - R1 -
- R1A - H11
-------------------------------------------------
<výpis vypuštěn>
-------------------------------------------------
Time = 299.208472580648
H11 : Ping 8 to 192.168.17.21
Path: H11 - R1A - R1 - R3 - R2 - R2B - H21 - H21 - R2B - R2 - R1 - R1A -
- H11
-------------------------------------------------
Time = 334.208472580648
H11 : Ping 9 to 192.168.17.21
Path: H11 - R1A - R1 - R2 - R2B - H21 - H21 - R2B - R2 - R1 - R1A - H11
-------------------------------------------------
```

V simulačním čase 19 sekund ještě směrovače neměly naplněné směrovací tabulky, směrovač R1A neznal cestu k počítači H21 (192.168.17.21), a proto byl ICMP paket zahozen. V simulačním čase 89 sekund jsou všechny linky funkční. ICMP paket je zaslán nejkratší cestou ze směrovače R1A, přes směrovače R1, R2, R2B, až k cílovému počítač H21. Cesta zpět je totožná.

ICMP paket zaslaný v čase 194 sekund se dostal stejnou cestou k počítači H21. V čase 200 sekund dojde pádu linky mezi směrovači R1 a R2. ICMP paket Echo-Reply je směrován zpět z H21 na směrovač R2. Směrovač R2 ještě nedostal novou informaci o síti 192.168.8.0/24. Tuto síť nezná a proto pošle počítači H21 nazpět ICMP paket typu Destination Unreachable (cíl není dostupný).

V čase 229 sekund je již ustanovena nová cesta. ICMP paket je ze směrovače R1 přeposlán na směrovač R3, následně na směrovač R2. V čase 299 sekund je linka mezi směrovači R1 a R2 stále nefunkční, proto ICMP paket putuje přes směrovač R3. V čase 300 sekund dojde k obnovení funkčnosti linky a zpět již putuje paket původní cestou, tedy ze směrovače R2 na směrovač R1.

Další test provedeme za použití ACL umístěného na směrovači R2. Modul ACL byl vytvořen v rámci bakalářské práce T. Suchomela [15]. ACL 110 je nastaven na portu eth2 směrovače R2 (ve směru od R3). Zápis v konfiguračním souboru:

```
   <ACLs>
 <ACL no="110">
   <entry seq_no="15">
```

```
      <action>deny</action>
      <IP_src>0.0.0.0</IP_src>
      <IP_dst>192.168.17.0</IP_dst>
      <WC_src>255.255.255.255</WC_src>
      <WC_dst>0.0.0.255</WC_dst>
      <protocol>icmp</protocol>
    </entry>
    <entry seq_no="20">
      <action>permit</action>
      <IP_src>0.0.0.0</IP_src>
      <IP_dst>0.0.0.0</IP_dst>
      <WC_src>255.255.255.255</WC_src>
      <WC_dst>255.255.255.255</WC_dst>
      <protocol>ip</protocol>
    </entry>
    <interfaces>
      <interface dir="in">eth2</interface>
    </interfaces>
  </ACL>
     </ACLs>
```

Výstup simulace je:

```
------------------------------------------------
Time = 159.208472580648
H11 : Ping 4 to 192.168.17.21
Path: H11 - R1A - R1 - R2 - R2B - H21 - H21 - R2B - R2 - R1 - R1A - H11
------------------------------------------------
Time = 194.208472580648
H11 : Ping 5 to 192.168.17.21
Path: H11 - R1A - R1 - R2 - R2B - H21 - H21 - R2B - R2 - R2 - R2B - H21 -
- ICMP ERROR
------------------------------------------------
Time = 229.208472580648
H11 : Ping 6 to 192.168.17.21
Path: H11 - R1A - R1 - R3
------------------------------------------------
Time = 264.208472580648
H11 : Ping 7 to 192.168.17.21
Path: H11 - R1A - R1 - R3
------------------------------------------------
Time = 299.208472580648
H11 : Ping 8 to 192.168.17.21
Path: H11 - R1A - R1 - R3
------------------------------------------------
Time = 334.208472580648
H11 : Ping 9 to 192.168.17.21
Path: H11 - R1A - R1 - R2 - R2B - H21 - H21 - R2B - R2 - R1 - R1A - H11
```

Počáteční a konečný stav výpisu je podobný jako při první simulaci. V čase 229 sekund, kdy je nefunkční linka mezi R1 a R2, je paket poslán přes směrovač R3. Ten předá paket směrovači R2, který ale paket díky ACL zahodí.

Závěrečný výpis statistiky použitého ACL (simulační čas 474 sekund):

```
IP datagrams received: 142
IP packets permitted without ACL action: 122
IP packets permitted by ACL action: 17
IP packets denied by an ACL action: 3
 - rule has been used: 3x
 - rule has been used: 17x
```

Z výpisu je zřejmé, že pravidlo zakazující vstup ICMP paketů bylo aplikováno třikrát.

5.2 Stabilita

Následující simulace se zaměří na stabilitu linky. I v této kapitole se zaměříme na linku mezi směrovači R1 a R2. Budeme vybírat různé časové intervaly mezi zapínáním a vypínáním rozhraní eth1 (dále jen Δt) na směrovači R2 (ve směru k R1), čímž budeme měnit stupeň nestability linky a sledovat chování sítě. Cílem simulace je zjistit, zda může nestabilita ovlivnit dostupnost sítě, i když máme záložní linky.

5.2.1 Simulace stability v nástroji Packet Tracer

Simulace této vlastnosti v PT by byla velmi náročná. Museli bychom vždy měřit čas, po kterém je třeba změnit stav linky, v tento čas zastavit simulaci a v CLI směrovače R2 vždy ručně vypnout či zapnout potřebné rozhraní. Avšak ani tato možnost není zcela proveditelná, protože PT je diskrétní simulátor. Díky tomu, že dochází ke krokovým změnám v čase při událostech, nelze provést zastavení simulace v námi požadovaný čas. Z tohoto důvodu jsme simulaci stability v PT neprováděli.

5.2.2 Simulace stability v nástroji OMNeT++

Pro tuto simulaci jsme použili manažer stavů linek [4]. Do souboru XML jsme zapsali časové odstupy mezi zapínáním a vypínáním rozhraní eth1 na směrovači R2. Postupně jsme volili Δt rovno 250, 100, 50, 25, 20, 10, 5 a 1 sekunda. Zatímco bude linka mezi směrovači R1 a R2 nestabilní, budeme testovat dostupnost mezi sítěmi 192.168.8.0/24 a 192.168.17.0/24 pomocí zasílání ICMP paketů.

Oproti simulaci dostupnosti jsme snížili čas mezi zasíláním ICMP paketů na 15 sekund, abychom mohli precizněji sledovat změny v dostupnosti sítě. Tento čas byl vybrán z toho důvodu, že odeslání ICMP Echo a příjem ICMP Echo-Reply trvá přes 10 sekund. Kdybychom použili kratší časovou prodlevu, výpisy simulace by byly chaotické a nepoužitelné.

V první sérii testů jsme nechali RIP konvergovat do času 100 sekund. V tomto čase jsou směrovací tabulky všech směrovačů v síti naplněny všemi cestami a síť je plně funkční. Po tomto čase došlo k vypínání a zapínání rozhraní ve výše uvedených intervalech. Simulace byla prováděna od času 0 po 250 sekund. Protože naměřené hodnoty byly pro všechny pokusy v rozmezí od 0 do 100 sekund stejné, tabulka 5.1 uvádí hodnoty naměřené až od simulačního času 100 sekund. Písmeno X v této i následujících tabulkách zastupuje simulaci sítě bez padajících linek.

Doba mezi změnami stavu linky (Δt) [s]	X	250	100	50	25	20	15	5	1
Počet změn stavu linky v čase 100 - 250 s	0	0	2	3	6	7	10	30	150
Počet doručených paketů	10	10	8	8	6	4	2	2	0
Počet nedoručených paketů	0	0	2	2	4	6	8	8	10
Paket jde běžnou cestu	10	10	3	3	3	2	2	2	0
Paket jde záložní cestou (přes R3)	0	0	5	5	3	2	0	0	0

Tabulka 5.1: Výsledky prvního měření v čase 100 - 250 sekund (posláno 10 paketů).

Dále jsme testovali, jak se výsledky změní pokud se nezačnou posílat ICMP pakety od času 0 sekund, ale později. Jako startovní čas pro pravidelné odesílání ICMP paketů jsme zvolili 3, 6, 9 a 12 sekund. Tabulka 5.2 ukazuje, jak se při těchto časových posunech měnil počet doručených paketů.

Doba mezi změnami stavu linky (Δt) [s]	100	50	25	20	15	5	1
Startovní čas v nula sekund	8	8	6	4	2	2	0
Startovní čas v třech sekundách	8	7	7	5	0	0	0
Startovní čas v šesti sekundách	8	7	7	5	0	0	0
Startovní čas v devíti sekundách	8	6	6	3	2	2	0
Startovní čas v dvanácti sekundách	8	6	5	5	3	0	0
Průměrný počet přijatých paketů	8	6,8	6,2	4,4	1,4	0,4	0

Tabulka 5.2: Počet doručených paketů v závislosti na startovním čase periodického zasílání ICMP paketů (simulační čas 100 - 250 sekund).

Časovač Trigger je zapnut v případě, že došlo k pádu některé linky nebo změně cesty. Po jeho uplynutí zašle směrovač informace o celé svojí směrovací tabulce všem svým sousedům. Délka tohoto intervalu se volí náhodně v rozmezí nula až pět sekund pro případ, že by v brzké době došlo k další změně. Zabrání se tak lavinovému zasílání zpráv v případě mnoha změn, které přicházejí rychle za sebou. V případě, že je linka vysoce nestabilní (interval mezi změnami je pod 10 sekund), hraje tento časovač významnou roli při informování sousedů o změnách směrovacích tabulek a tím pádem i v množství doručených paketů.

Stejně tak ovlivňuje množství doručených paketů v případě, že posouváme startovní čas odesílání ICMP paketů. V případě $\Delta t = 5$ je to nejzřetelnější (tabulka 5.2). Uvažujme, že v čase nula by byly naplněny směrovací tabulky. Kdyby v čase pět sekund spadla linka, do času šest sekund, tj. do jedné sekundy po pádu linky, se s největší pravděpodobností nestihnou zaslat aktualizace (časovač Trigger by musel být nastaven na čas menší než jedna sekunda). V čase devět sekund, tj. čtyři sekundy po pádu linky, jsou aktualizace rozeslány s mnohem větší pravděpodobností (Trigger by musel být nastaven v rozmezí nula až čtyři sekundy), proto při tomto posunu došlo k příjmu nejvíce, tj. dvou, paketů.

Po té jsme provedli podobnou sérii simulací, s tím rozdílem, že linka byla nestabilní už od zahájení simulace, tedy od simulačního času $0 + \Delta t$. Δt nabývala postupně hodnot 50, 25, 20, 10, 5 a 1 sekunda. $\Delta t = 100$ a 250 sekund jsme vynechali, protože pro tyto časy by byla situace stejná jako v předešlé sérii simulací. Frekvence zasílání ICMP paketů zůstala

stejná. Tabulka 5.3 uvádí hodnoty naměřené simulací od simulačního času 0 po 250 sekund.

Doba mezi změnami stavu linky (Δt) [s]	X	50	25	20	10	5	1
Počet změn stavu linky v čase 0 - 250 s	0	5	10	12	25	50	250
Počet úspěšně doručených paketů	12	12	9	8	6	2	0
Počet nedoručených paketů	4	4	7	8	10	14	16
Paket jde běžnou cestu	12	6	5	5	2	2	0
Paket jde záložní cestou (přes R3)	0	6	4	3	4	0	0

Tabulka 5.3: Výsledky druhého měření v čase 0 - 250 sekund (posláno 16 paketů).

I když je síť stabilní (X), dojde ke ztrátě čtyř paketů. Tato ztráta je způsobena tím, že na počátku simulace směrovač R1A nezná cestu do sítě 192.168.17.0/24, a proto pakety zahodí.

Následující tabulka 5.4 uvádí časy, ve kterých došlo k naplnění směrovací tabulky směrovače R1A cestou do sítě 192.168.17.0/24 a tím pádem získání potenciální možnosti směrovat ICMP pakety. Měření jsme prováděli prostým sledováním směrovacích tabulek při simulaci. Ve chvíli, kdy se objevila cesta v tabulce směrovače R1A, zaznamenali jsme čas do tabulky.

Doba mezi změnami stavu linky (Δt) [s]	X	50	25	20	10	5	1
Simulační čas získání cesty k cílové síti [s]	60	60	30	30	30	22	12

Tabulka 5.4: Simulační časy, ve kterých byla získána cesta do sítě 192.168.17.0/24.

5.3 Shrnutí kapitoly

V této kapitole jsme si předvedli, jak jde simulační modely vytvořené podle návrhových vzorů firmy Cisco využít k simulaci dostupnosti a stability linek. Vybrali jsme typické vlastnosti sítě, které se v praxi ověřují, neboť jsou klíčové pro správnou funkčnost sítě. Pokud to bylo možné, provedli jsme simulaci vybraných vlastností v nástrojích OMNeT++ a PT. Výsledky jsme si předvedli na výpisech simulátorů. V případě simulace stability by se jednalo o mnoho dlouhých výpisů, proto jsme výsledky shrnuli v přehledných tabulkách.

Kapitola 6

Výsledky simulací v nástrojích OMNeT++ a Packet Tracer

V této kapitole porovnáme výsledky simulací v nástrojích OMNeT++ a Packet Tracer. Zhodnotíme simulační možnosti obou nástrojů a nakonec se zaměříme na přínos simulace pro praktickou analýzu počítačových sítí.

6.1 Výsledky simulací

6.1.1 Výsledky simulace dostupnosti

Při simulaci dostupnosti jsme provedli dvě simulace. Nejprve jsme zjišťovali, co se stane, pokud spadne linka mezi R1 a R2. Dle předpokladů byla využita záložní cesta přes směrovač R3. Ukázalo se, že směrovače nebyly schopné okamžitě přesunout provoz na záložní linku. V nejhorším případě musíme vyčkat přibližně 30 sekund, což je implicitní délka časovače Update. To je případ, kdy se aktualizační zprávy posílaly těsně před pádem linky. Obvykle je ale tento čas nižší. Dokázali jsme tím, že protokol RIP konverguje pomalu.

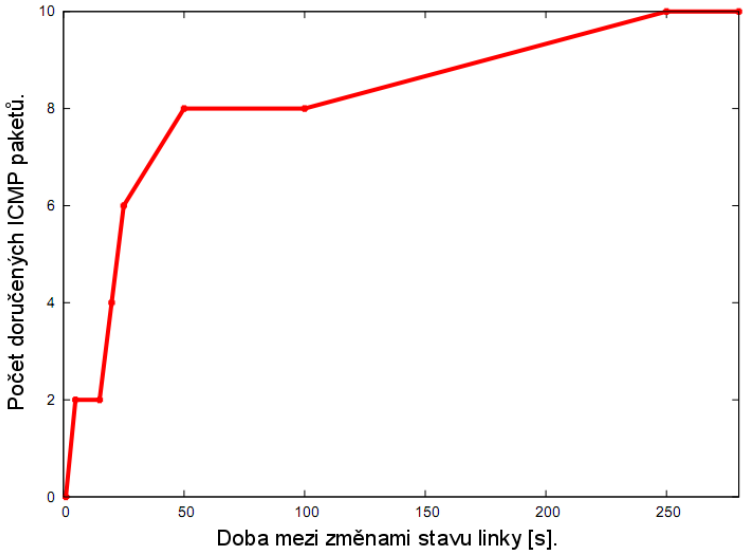
Při druhé simulaci jsme využili ACL na rozhraní eth2 směrovače R2, které blokovalo všechny ICMP pakety směřující do sítě 192.168.17.0/24. Po pádu linky přesměrovali směrovače znovu provoz na směrovač R3. Ten odeslal ICMP pakety na směrovač R2 stejně jako v předchozí simulaci. ACL na směrovači R2 ale tyto pakety zahazoval. Síť 192.168.17.0 /24 se stala nedostupnou.

Touto simulaci jsme chtěli ukázat, že i když máme pro provoz mezi sítěmi LAN záložní linku a předpokládáme, že bude využita v případě poruchy hlavní linky, nemusí to být vždy pravda. Může se stát, že vlivem dalších parametrů, které jsme si neuvědomili (jako je nastavené ACL), se mohou stát sítě nedostupné. Z tohoto důvodu je vhodné provést podobnou simulaci pro vlastní síť. Výsledky simulace dostupnosti v nástrojích OMNeT++ a Packet Tracer byly shodné.

6.1.2 Výsledky simulace stability

Při simulace stability sítě jsme se zaměřili na zkoumání množství ICMP paketů, které byly v pořádku doručeny zpět počítači H11. Naměřené hodnoty z tabulky 5.1 jsou znázorněny v grafu 6.1. Síť divergovala. Se zvyšující stabilitou linky se zvyšoval počet doručených paketů až dosáhl počtu všech odeslaných paketů. Porovnáním tabulek 5.1 a 5.3 zjistíme, že síť se chovala obdobně, ať už byla linka nestabilní od počátku, nebo až po naplnění směrovacích

tabulek. Při nestabilitě linky se zátěž běžné a záložní linky rozdělila přibližně na polovinu. Průměrné hodnoty z tabulky 5.2 můžeme vidět v grafu 6.2. Z tohoto grafu je zřejmé, že při Δt menším než 25 sekund se počet doručených paketů rapidně snižoval. Směrovače si totiž nemohli korektně zasílat aktualizace, které se zasílají jednou za 30 sekund.



Obrázek 6.1: Graf závislosti počtu přijatých paketů na Δt dle tabulky 5.1.

Nastavení časovače Trigger má také vliv na počet doručených paketů zmíněný v předchozí kapitole, viz tabulka 5.2. Tento časovač se ale podle RFC 1058 [5] nedá nastavit napevno, nýbrž se nastavuje náhodně. Proto, pokud se data nebudou posílat pravidelně jako v naší simulaci, je počet přenesených paketů za čas víceméně věcí náhody.

Dále jsme došli k zajímavému zjištění, že se zvyšující se nestabilitou (zmenšujícím se časem mezi změnami stavu linky) sítě se snižuje čas, ve kterém směrovač R1A získá cestu do sítě 192.168.17.0/24 (tabulka 5.4). Nastíníme si situaci u stabilní sítě. V čase 0 sekund rozešlou směrovače sousedům informace o přímo připojených sítích. Při dalším zasílání aktualizací v čase 30 sekund už zasílají směrovače nejen své přímo připojené sítě, ale i cesty od svých sousedů, proto se v tomto čase dozví směrovače R1 o síti 192.168.17.0/24. Směrovači R1A tuto informaci předá až při další aktualizaci v čase 60 sekund.

Časovač Update není nastaven přesně na 30 sekund, ale k tomuto času se přidává krátká prodleva (je součástí implementace) v rozmezí nula až jedna sekunda, aby všechny směrovače nezasílaly aktualizace ve stejný čas. Tím pádem není předem jasné, který směrovač dříve zašle informaci sousedovi. Pokud by v simulačním čase 30 sekund nejprve získal směrovač R1 informaci o cestě do sítě 192.168.17.0/24 a až následně zaslal směrovací informace směrovači R1A, mohly by směrovač R1A znát cesty do všech sítí již v tomto čase. Takovýto stav by mohl ale nastat až později při simulaci, kdy se rozestupy mezi zasíláním aktualizací jednotlivými směrovači zvětší. Na počátku simulace odešle směrovač R1 směrovači R1A aktualizace dříve, než zpracuje příchozí informace.

Podobně jako v případě stabilní sítě se síť chová i tehdy, je-li Δt větší než 30 sekund. Je-li ale tento čas nižší, situace se mění. Vezměme si jako příklad $\Delta t = 20$ sekund. V čase nula se směrovač R3 dozví přímé sítě od směrovačů R2 a R1, síť 192.168.17.0/24 nezná. V čase 20 sekund dojde k výpadku linky mezi R1 a R2. Oba směrovače začnou zasílat svým

Obrázek 6.2: Graf závislosti průměrného počtu přijatých paketů na Δt dle tabulky 5.2.

sousedům své aktualizace. Směrovač R3 se tak dozví od R2 o síti 192.168.17.0/24. V čase 30 sekund dojde k zasílání pravidelných aktualizací všem svým sousedům a tak se směrovač R1, následně i R1A, dozví od směrovače R3 o existenci sítě 192.168.17.0/24.

Ještě zajímavější situace nastává při intervalu $\Delta t = 5$ sekund. V čase nula se směrovače dozví přímé sítě svých sousedů. Směrovač R2 se dozví o síti 192.168.17.0/24. V čase 5 sekund dojde k pádu linky. V čase 10 sekund dojde k zprovoznění linky. Při této změně směrovače začnou zasílat celé své tabulky všem svým sousedům, R2 však nestihne zaslat informaci směrovači R1 z důvodu delšího intervalu časovače Trigger. V čase 20 sekund, kdy dojde znovu k obnově linky, již R2 zašle směrovači R1 informaci o síti 192.168.17.0/24. R1 zašle zprávu o této cestě ihned směrovači R1A.

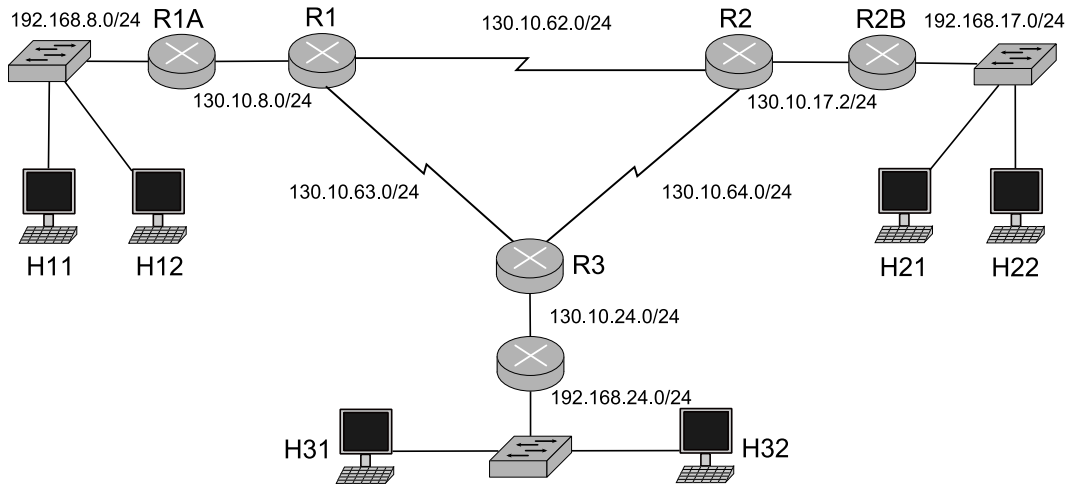
6.2 Porovnání nástrojů OMNeT++ a Packet Tracer

Pro simulace jsme použili nástroje OMNeT++ i PT. Simulováním jsme ověřili, že oba nástroje neposkytují stejné simulační možnosti. V praxi se ukázalo, že Packet Tracer je vhodnější jako nástroj pro modelování sítí než pro jejich simulaci. Za to nástroj OMNeT++ prokázal, že je ryzím simulačním nástrojem.

Simulace v OMNeT++ byla jednoduchá hlavně díky manažeru stavů linek [4]. Do XML souboru jsme popsali, jak chceme aby simulace probíhala. Díky manažeru bylo možné provést simulaci pádu linek v jakémkoliv simulačním čase, s jakoukoliv frekvencí a hlavně v jakémkoliv množství. Při simulaci stability sítě jsme mohli simulovat všechny stupně nestability (všechny Δt) při jedné simulaci. Pro větší přehlednost jsme ale této možnosti nevyužili. Abychom získali vhodný výstup, ukazující jakou cestou byl paket směrován, bylo třeba dopsat pár řádků kódu do jednoduchých modulů představujících síťovou vrstvu (Network Layer). Informace o průběhu simulace se pak vypisovaly do konzolového okna, ze kterého se daly jednoduše překopírovat a uchovat pro další použití.

Při simulaci v PT se informace o průběhu simulace vypisují do okna Even List (obrázek 1.6) a není možné je nijak exportovat, což je velmi nepraktické. Proto jsme museli využít

snímání obrazovky (Print Screen) a do této práce vložit výsledky jako obrázek. Hlavní nevýhoda PT ale tkví v tom, že není možné nijak naplánovat simulaci. Narozdíl od OMNeT++ není ani možné doprogramovat modul, který by simulaci ovládal. V případě, že chceme simulovat pády a obnovení linky, je nutné vždy simulaci zastavit a ručně zapsat příkaz pro zapnutí/vypnutí rozhraní pro každý směrovač a rozhraní zvlášť. Simulování nestability linky je proto v tomto nástroji nemožné.



Obrázek 6.3: Model sítě, kterou využíváme v simulacích.

Přepokládejme, že bychom chtěli simulovat dostupnost z jedno PC na všechna další PC v síti. V naší síti (obrázek 6.3) máme šest PC ($n = 6$), každé PC by zasílalo pakety dalším pěti PC. To znamená, že bychom museli provést 5 x 6 simulaci. Pro obecný počet PC je to

$$(n-1)n = n^2 - n, \tag{6.1}$$

což vede na kvadratickou složitost $O(N^2)$. Dále bychom zkoumali, jak se bude síť chovat v případě pádu jakékoliv jedné linky po cestě mezi dvěma PC. V naší topologii to znamená sedm linek ke každému ze čtyř počítačů v jiné síti LAN a dvě linky k počítači ve stejné síti LAN. To je v průměru šest linek. Tuto hodnotu označíme jako r. V našem případě $r = 6$, tedy $r = n$. Hodnotou r musíme vynásobit počet simulací dostupnosti mezi dvěma PC:

$$(n^2 - n)r \tag{6.2}$$

Složitost simulace můžeme vypočítat následovně:

$$(n^2 - n)n = n^3 - n^2 \tag{6.3}$$

Simulace dostupnosti všech sítí v PT vede na kubickou složitost $O(N^3)$. Použijeme-li vzorec 6.2 můžeme vypočítat, že pro naší topologii bychom museli provést celkem 180 simulací. V případě, že bychom se rozhodli simulovat nefunkčnost všech možných variací linek po cestě (kromě situace, kdy by byly nefunkční všechny linky na trase), musíme provést

$$2^n - 1 \tag{6.4}$$

simulací pro jednu dvojici počítačů. V případě, že bychom simulovali také dostupnost mezi všemi počítači v síti, bylo by třeba provést

$$(n^2 - n)(2^n - 1) \tag{6.5}$$

simulací. Tím pádem se dostáváme k exponenciální složitosti $O(2^N)$. Pro naší topologii bychom museli provést 1890 simulací. Z tohoto důvodu můžeme prohlásit PT za nevhodný nástroj pro praktickou simulaci počítačových sítí. V nástroji OMNeT++ je pouze nutné zopakovat simulaci pro každou dvojici počítačů, tedy provést 30 simulací (podle vzorce 6.1).

6.3 Přínos simulace pro praktickou analýzu počítačových sítí

Jak jsme si v předchozí sekci dokázali, nástroj PT nelze pro praktickou analýzu počítačových sítí použít. Nástroj OMNeT++ má v tomto směru mnohem lepší vyhlídky.

Implementací protokolu RIP a redistribuce jsme výrazně rozšířili možnosti tohoto nástroje. Společně s OSPF [4], EIGRP [16], ACL [15], manažeru linek [4] a Cisco směrovače [14] se z obecného diskrétního simulátoru stal plnohodnotný simulátor počítačových sítí. Díky konfiguračnímu souboru [13] podobnému CLI je simulátor připraven na modelování a simulaci sítí složených z Cisco směrovačů.

Rozhodneme-li se otestovat různé vlastnosti sítě, je využití simulátoru OMNeT++ se všemi výše jmenovanými doplňkovými modely rozhodně dobrým řešením. Není nutné narušovat chod velké sítě testovacími ACL či pády linek. Některé vlastnosti se na reálné síti testují jen velmi těžce. Rozhodneme-li se testovat stabilitu sítě, neexistuje jednoduchý způsob, kterým bychom nasimulovali nestabilitu na reálné síti. Při použití simulátoru se nemůže nic pokazit ani zničit, můžeme tedy simulátor považovat za nejlevnější variantu pro testování sítě.

Většina simulátorů uvedených v sekci 1.1 se používá k simulování různých věci, nejen sítí, proto má většina těchto simulátorů omezené nebo žádné možnosti směrování. Naopak OMNeT++ je připraven na simulaci běžných Cisco sítí. Protokol RIP byl naimplementován podle příslušného RFC a tak je plně kompatibilní s implementaci nejen v běžných směrovačích. Sítě v OMNeT++ se chovají podobně jako sítě reálné.

6.4 Shrnutí kapitoly

V této kapitole jsme si shrnuli a odůvodnili výsledky, ke kterým jsme došli při simulaci v předešlé kapitole. V případě dostupnosti jsme ukázali, že použitím simulace je možné odhalit některé parametry, které mohou ovlivnit dostupnost sítě. Prokázali jsme, že dostupnost sítě také závisí na stabilitě linek. Přestože při výpadku linky se nabízelo použití záložní linky, protokol RIP ji nedokázal využít. Tento protokol totiž konverguje příliš pomalu, a tak po výpadu linky nedokázal ihned přesměrovat data na záložní linku. Tímto jsme dokázali, že záložní linka není vždy zárukou dostupnosti sítí a protokol RIP není vhodným směrovacím protokolem pro nestabilní sítě. Dokázali jsme také, že se snižujícím se časem mezi změnami stavu linek se zkracuje čas získání směrovacích informací o všech sítích.

Porovnali jsme nástroje PT a OMNeT++ z hlediska možností při simulacích sítí. Ukázalo se, že použití nástroje PT pro hlubší simulace námi namodelovaných topologií, je prakticky nemožné. Simulovat v něm nestabilní linky je neproveditelné. Při simulaci dostupnosti v PT se můžeme při provádění simulací pro jednotlivé situace dostat ke kubické nebo až exponenciální složitosti. Na závěr jsme zhodnotili možnosti nástroje OMNeT++ pro praktickou analýzu počítačových sítí. Ukázalo se, že implementací protokolu RIP, jsme výrazně obohatili nástroj v oblasti simulace směrování sítí.

Závěr

V rámci této práce jsme detailně prostudovali směrovací protokoly RIP a OSPF, nalezli návrhové vzory firmy Cisco, které pracují s těmito protokoly, seznámili se s nástroji PT a OMNeT++ a jejich možnostmi na poli modelování a simulace. Cílem této práce bylo vytvořit simulační modely v praxi použitelných sítí v těchto nástrojích a simulováním modelů předvést možnosti nástrojů pro praktickou analýzu počítačových sítí.

Vlastní přínos

Nejprve jsem se seznámila s modelovacími a simulačními možnostmi nástrojů PT, OMNeT++ a framework INET pročtením dostupných manuálu a vyzkoušením řady simulací. Ukázalo se, že pro modelování sítí využívajících Cisco zařízení je vhodnější PT od téže firmy. Modelování v OMNeT++ je náročnější díky nutnosti naučit se programovací jazyk NED a C++ a především díky nedostatečným informacím k framework INET.

Na základě návrhových vzorů pro směrování firmy Cisco jsem vytvořila simulační modely v nástroji PT. Abych mohla vytvořit stejné modely i pro nástroj OMNeT++, bylo nejdříve nutné naimplementovat moduly, které se nenacházely v knihovně OMNeT++ ani v framework INET. Jednalo se protokol RIP a redistribuci mezi protokolem OSPF a protokolem RIP.

Po pečlivém nastudování RFC 1058 jsem naimplementovala v jazyce C++ chování modulu RIPRouting, který zajišťuje směrování na základě protokolu RIP. Podobně jsem doimplementovala i redistribuci z protokolu OSPF do protokolu RIP. Pro modul jsem také vytvořila popis v jazyce NED. Poté, co jsem chybějící vlastnosti naimplementovala, vytvořila jsem v jazyce NED simulační modely také pro nástroj OMNeT++ a provedla jsem nastavení všech směrovačů v XML konfiguračních souborech.

Když jsem měla připraveny všechny simulační modely, definovala jsem si vlastnosti, které budu simulovat. Některé třídy z framework INET jsem doplnila o kód, který umožňuje vypisovat trasy ICMP paketu. Vybrala jsem si simulační model využívající směrovací protokol RIP a provedla jsem simulace dostupnosti a stability v obou simulačních nástrojích. Pro simulace v OMNeT++ jsou vytvořila řadu simulačních scénářů v jazyce XML. Výsledky simulací jsou zaznamenány v příslušné kapitole.

Zhodnotila jsem výsledky simulací. Ukázala jsem, že při zkoumání dostupnosti se mohou naskytnout jevy, se kterými jsme na počátku nepočítali (ACL). Dále jsem ukázala, že nestabilita linky má negativní vliv na dostupnost linky a pozitivní vliv na rychlost získání počátečních směrovacích informací. Při těchto simulacích se nástroj PT ukázal jako nevhodný simulační nástroj. Simulaci je totiž třeba pro každý pád linky opakovat, tento postup dosahoval až exponenciální složitosti.

Naopak OMNeT++ obohacený o mou implementaci dokáže plnohodnotně simulovat sítě využívající směrovací protokoly RIP a OSPF. Může tak pomoci při praktické analýze

sítí a v případě nestabilních linek se může stát jednou z mála možností, jak síť otestovat.

Další vývoj

Tato práce vznikla v rámci projektu ANSA [26], který se zaměřuje na automatizované metody verifikace sítí založené na konfiguraci síťových zařízení a síťové topologie. Tato práce se může v tomto směru dále rozvíjet a projektu nabídnout další funkce a moduly pro automatizovanou simulací sítě.

V případě protokolu RIP by bylo možné implementaci rozšířit o autosumarizaci, která pro uváděné simulace nebyla potřeba, a také o Poison Reverse, který by se mohl volitelně použít místo implementovaného Split Horizon. V rámci této práce byl naimplementován pouze protokol RIP verze jedna, dále by se implementace mohla rozšířit o verzi dvě.

Pro simulaci jako takovou by bylo vhodné napsat třídu, která by dokázala vypisovat informace, o tom, co se v síti komu zasílá. Bylo by to vhodnější, než zásah do existujících tříd z framework INET, který jsem kvůli simulačním výpisům provedla já.

Literatura

[1] Internetwork Design Guide: RIP and OSPF Redistribution [online]. [cit. 2009-02-13]. URL http://www.cisco.com/en/US/docs/internetworking/design/guide/nd2014.pdf

[2] *CCNA Exploration 4.0: Routing Protocols and Concepts [online]*. 2007, Přístupný pro studenty a vyučující kurzu CCNA2 po přihlášení na. URL http://www.cisco.com/web/learning/netacad/index.html

[3] *Packet Tracer 5.0 Help*. 2008, součást instalace programu Packet Tracer 5.0.

[4] Danko, M.: *Modelování směrovacích protokolů OSPF v simulátoru OMNeT++*. Bakalářská práce, FIT VUT v Brně, 2009.

[5] Hedrick, C.: *RFC 1058: Routing Information Protocol*. 1988. URL http://tools.ietf.org/html/rfc1058

[6] Hedrick, C.: *An Introduction to IGRP*. 1991. URL http://www.cisco.com/en/US/tech/tk365/technologies_white_paper09186a00800c8ae1.shtml

[7] Lomnický, M.; Veselý, V.: Směrování a směrovací protokoly [online]. 2007, [cit. 2008-11-20]. URL http://netacad.fit.vutbr.cz/texty/ccna-moduly/ccna2-6.pdf

[8] Malkin, G.: *RFC 2453: RIP Version 2*. 1998. URL http://tools.ietf.org/html/rfc2453

[9] Moy, J.: *RFC 2328: OSPF Version 2*. 1998. URL http://tools.ietf.org/html/rfc2328

[10] Oran, D.: *RFC 1142: OSI IS-IS Intra-domain Routing Protocol*. 1990. URL http://tools.ietf.org/html/rfc1142

[11] Peringer, P.: Modelování a simulace IMS, studijní opora. 2006.

[12] Ryšavý, O.: Network Modeling and Simulation [online]. 2008, [cit. 2008-11-20]. URL https://wis.fit.vutbr.cz/FIT/st/course-files-st.php/course/ISA-IT/lectures/isa11-simulace.pdf

[13] Scherfel, P.: *Simulace chování sítě na základě analýzy konfiguračních souborů aktivních síťových zařízení*. Bakalářská práce, FIT VUT v Brně, 2009.

[14] Sivák, V.: *Model Cisco směrovače v simulačním nástroji OMNeT++*. Bakalářská práce, FIT VUT v Brně, 2009.

[15] Suchomel, T.: *Rozšíření simulátoru OMNeT++ o filtrovací pravidla ACL*. Bakalářská práce, FIT VUT v Brně, 2009.

[16] Tlolka, M.: *Simulace EIGRP protokolu v prostředí OMNeT++*. Bakalářská práce, FIT VUT v Brně, 2009.

[17] Varga, A.: *OMNeT++ Descrete Event Simulation System. User manual version 4.0*. 2008.

[18] WWW stránky: About GloMoSim. [cit. 2009-04-25].
URL http://pcl.cs.ucla.edu/projects/glomosim/

[19] WWW stránky: The CNET network simulator. [cit. 2009-04-25].
URL http://www.csse.uwa.edu.au/cnet/

[20] WWW stránky: Erlang Traffic and Queuing Software. [cit. 2009-04-25].
URL http://members.iinet.net.au/~clark/

[21] WWW stránky: GTNetS. [cit. 2009-04-25].
URL http://www.ece.gatech.edu/research/labs/MANIACS/GTNetS/

[22] WWW stránky: JiST - Java in Simulation Time / SWANS. [cit. 2009-04-25].
URL http://jist.ece.cornell.edu/

[23] WWW stránky: OMNeT++ Community Site. [cit. 2009-04-25].
URL http://www.omnetpp.org/

[24] WWW stránky: Packet Tracer. [cit. 2009-04-25].
URL http://www.cisco.com/web/learning/netacad/course_catalog/PacketTracer.html

[25] WWW stránky: Performance Prophet - A Performance Modeling and Prediction System for Cluster and Grid Computing. [cit. 2009-04-25].
URL http://www.dps.uibk.ac.at/projects/prophet/

[26] WWW stránky: Project ANSA, Brno University of Technology. [cit. 2009-05-10].
URL https://nes.fit.vutbr.cz/ansa/pmwiki.php

[27] WWW stránky: Scalable Network Technologies: Creators of QualNet Simulator. [cit. 2009-04-25].
URL http://www.scalable-networks.com/

[28] WWW stránky: Tetcos software - NetSim. [cit. 2009-04-25].
URL http://www.tetcos.com/software.html

[29] WWW stránky: UCLA Parsec Programming Language. [cit. 2009-04-25].
URL http://pcl.cs.ucla.edu/projects/parsec/

Příloha A

Obsah DVD

V následující tabulce A.1 je uveden obsah přiloženého DVD.

doc\	Programová dokumentace v formátu HTML vygenerovaná programem Doxygen.
INET\	Rozbalená a přeložená knihovna INET i s upravenými soubory, doimplementovanými knihovny a vlastními příklady simulací. V tomto stavu se využívala pro tuto práci.
instalation\	Instalační soubory pro PT, OMNeT++ a INET.
models omnet\	Simulační modely pro OMNeT++. Jsou také vloženy ve struktuře složky INET.
models PT\	Simulační modely pro PT.
rip\	Třídy implementující protokol RIP a redistribuci. Jsou také vloženy ve struktuře složky INET.
tex\	Zdrojové soubory této práce psané v LaTeX.
doc.chm	Programová dokumentace v formátu CHM vygenerovaná programem Doxygen.
projekt.pdf	Elektronická verze této práce v formátu PDF.
readme.txt	Obsah DVD.

Tabulka A.1: Obsah přiloženého DVD.

Příloha B

Instalační příručka

Tato instalační příručka by vám měla pomoci s instalací simulačních nástrojů, které se využívali v rámci této práce. Všechny simulace byly prováděny pod operačním systémem Windows XP, proto jsou zde uvedeny jen instalační postupy používané pod tímto operačním systémem.

B.1 Instalace programu PacketTracer 5.0

Packet Tracer je simulační nástroj od firmy Cisco volně dostupný pro studenty a instruktory Cisco Networking Academy. Ti si mohou po přihlášení na stránkách firmy Cisco stáhnout instalátor nejnovější verze Packet Tracer 5.0 [24] (dále jen PT). Na výběr je instalátor obsahující pouze samotný program, verze s mnoha demo příklady, která je vhodná především pro začínající uživatele, či verze obsahující instalátor společně s tutoriály. Je zde dostupných i několik dokumentů ve formátu `.pdf`, které ale mají pouze obecný seznamovací charakter a uživatel se s v nich nedozví žádnou podstatnou informaci, která by mu zjednodušila užívání programu.

Nejnovější verze PT je distribuována jak pro operační systém Windows, tak pro Linux. Verze pro Linux jsou ke stažení tři – konkrétně pro Ubuntu 7.10, Fedoru 7 a pro Linux s jádrem verze 2.6 a vyšší.

Postup instalace:

1. Spustíme soubor `PacketTracer5_setup.exe`.
2. Přečteme si licenční ujednání (EULA) a odsouhlasíme jej.
3. Vybereme cíl instalace. Implicitně je nastaveno na složku `C:\Program Files\Packet Tracer 5.0`.
4. Následně jsme dotázání, jestli si přejeme vytvořit složku v Start menu.
5. Nakonec se nás program zeptá, jestli si přejeme vytvořit ikonu na ploše (desktop icon) nebo ikonu v panelu snadného spuštění (quick lunch icon).
6. Následuje samotná instalace.

B.2 Instalace nástroje OMNeT++

Instalační soubor pro Windows nebo balík pro systémy UNIX lze stáhnout, ze stránek OMNeT++ [23]. Pro Windows stáhněte archiv `omnetpp-4.0-src-windows.zip` (161 MB) a rozbalte ho. Instalace obsahuje kromě knihovny OMNeT++ také IDE a MinGW, který umožní instalaci pod OS Windows.

Postup instalace:

1. Spustíme instalační prostředí s bash shell `mingwenv.cmd`.
2. Provedeme kontrolu nastavení pomocí `configure`.
3. Příkazem `make` provedeme překlad.
4. Zkontrolujeme funkčnost spuštěním vzorových ukázek. Přejdeme do složky s příklady pomocí `cd samples/dyna` a spustíme je příkazem `./dyna`.
5. IDE můžeme spustit z příkazové řádky příkazem `omnetpp`.

B.3 Instalace framework INET

Nejnovější verze framework INET má název INETMANE 20080920. Pro jeho správnou funkčnost je nutné mít nainstalovaný OMNeT++ 4.0. Ze stránek OMNeT++ stáhněte soubor `INETMANET-20080920.tbz2` a rozbalte jej.

Import do IDE:

1. Otevřeme IDE.
2. V menu vyberte položku `File -> Import...`
3. V okně se nalézá stromová struktura. V ní vyberte `General -> Existing Projects into Workspace` a stiskněte tlačítko `Next`.
4. V následujícím okně ponechte zatrženou možnost `Select Root Directory` a pomocí volby `Browse...` nalezněte složku, do které jste rozbalili INET. Zatrhněte INET project a stiskněte tlačítko `Finish`.
5. INET je nyní součásti nového projektu.

Příloha C

Implementace RIPRouting

V této příloze uvádíme kompletní hlavičkový soubor k třídě RIPRouting (bez vložených knihoven).

```
// Struktura představuje interní tabulku rozhraní.
struct RIPinterface
{
   int  intID;              // Identifikátor rozhraní.
   IPvXAddress  addr;       // IP adresa na rozhraní.
   IPvXAddress  mask;       // Maska na rozhraní.
   bool  broadcast;         // Umožňuje broadcast?
   bool  loopback;          // Jde o loopback?
   bool passive;            // Můžou se na rozhraní zasílat RIP zprávy ?
   InterfaceEntry*entry;    // Struktura popisující rozhraní.
};

// Struktura spojuje cestu ze směrovací tabulky s časovačem protokolu RIP.
struct RIPRouteTimer
{
   IPRoute *  route;        // Cesta ze směrovací tabulky.
   RIPTimer *  timer;       // Časovač přiřazený k cestě.
};

// Struktura zaznamenává dvě různé cesty do jedné sítě.
struct RIPRouteTwins
{
   IPRoute *  route1;
   IPRoute *  route2;
};

// Struktura nese podstatné informace o redistribuci.
struct RIPRedistribution
{
   char * protocol;           // Redistribuovaný protokol (např. OSPF).
   int  metric;               // Metrika nových RIP cest.
   bool redistrinute;         // Je zapnuta redistribuce?
```

```
};
class RIPRouting: public UDPAppBase, protected INotifiable {
   private:
      std::vector<RIPRouteTimer>routeTimer;
      std::vector<IPAddress>   network;
      std::vector<RIPinterface>  ripIft;
      std::vector<IPRoute *>intDown;
      std::vector<RIPRouteTwins> routeTwins;
      RIPTimer * triggerTimer;
      IRoutingTable            *rt;
      IInterfaceTable          *ift;
      NotificationBoard  *nb;
      UDPControlInfo           *udpCtrl;
      int                      localPort;
      int                      destPort;
      const char * hostname;
      RIPRedistribution redistr;

      // zpracovaní RIP zpráv
      void processPacket(cMessage *msg);
      void processRequest(RIPPacket *msg);
      void processResponse(RIPPacket *msg);

      // odesílání a tvorba RIP zpráv
      void sendPacket(int command, IPAddress destAddr);
      RIPPacket* createPacket(int command, InterfaceEntry * entry);
      std::vector<RouteEntry> fillNetworks(InterfaceEntry * entry);

      // nacteni konfigurace, redistribuce, notifikace
      bool LoadConfigFromXML(const char * filename);
      std::vector<RouteEntry> getOSPFRoutes();
      void receiveChangeNotification(int category, const cPolymorphic *details);

      // pomocné metody
      void sendTrigger();
      bool checkTwin(IPRoute * entryRT);
      RIPTimer * updateTimer(int type, RIPRouteTimer * entry);
      int getRouteRT (RIPTimer *timer);
      int getTimerRT (IPRoute *route);
      void insertIft(InterfaceEntry * entryIFT);

   protected:
      virtual int numInitStages() const  {return 4;}
      virtual void handleMessage(cMessage *msg);
      virtual void initialize(int stage);
   public:
      virtual ~RIPRouting();
};
```

Příloha D

Konfigurační soubory

D.1 Konfigurační soubor pro OSPF model

Uvedený model je použit u simulačního modelu, který využívá směrovací protokol OSPF a rozdělení sítě do čtyř OSPF oblastí. Uvádíme jen konfiguraci pro směrovač R1.

```
<Router id="130.10.63.1"> <!-- R1 -->
  <Interfaces>
   <Interface name="eth0">
      <IPAddress>130.10.8.1</IPAddress>
      <Mask>255.255.255.0</Mask>
      <OspfNetworkType>point-to-point</OspfNetworkType>
      <OspfCost>1</OspfCost>
      <OspfHelloInterval>10</OspfHelloInterval>
      <OspfDeadInterval>40</OspfDeadInterval>
      <OspfRetransmissionInterval>5</OspfRetransmissionInterval>
      <OspfInterfaceTransmissionDelay>1</OspfInterfaceTransmissionDelay>
      <OspfAuthenticationType>Null</OspfAuthenticationType>
      <OspfAuthenticationKey>0x00</OspfAuthenticationKey>
    </Interface>
    <Interface name="eth1">
      <IPAddress>130.10.62.1</IPAddress>
      <Mask>255.255.255.248</Mask>
      <OspfNetworkType>point-to-point</OspfNetworkType>
      <OspfCost>1</OspfCost>
      <OspfHelloInterval>10</OspfHelloInterval>
      <OspfDeadInterval>40</OspfDeadInterval>
      <OspfRetransmissionInterval>5</OspfRetransmissionInterval>
      <OspfInterfaceTransmissionDelay>1</OspfInterfaceTransmissionDelay>
      <OspfAuthenticationType>Null</OspfAuthenticationType>
      <OspfAuthenticationKey>0x00</OspfAuthenticationKey>
    </Interface>
    <Interface name="eth2">
      <IPAddress>130.10.63.1</IPAddress>
      <Mask>255.255.255.248</Mask>
      <OspfNetworkType>point-to-point</OspfNetworkType>
```

```
      <OspfCost>1</OspfCost>
      <OspfHelloInterval>10</OspfHelloInterval>
      <OspfDeadInterval>40</OspfDeadInterval>
      <OspfRetransmissionInterval>5</OspfRetransmissionInterval>
      <OspfInterfaceTransmissionDelay>1</OspfInterfaceTransmissionDelay>
      <OspfAuthenticationType>Null</OspfAuthenticationType>
      <OspfAuthenticationKey>0x00</OspfAuthenticationKey>
    </Interface>
  </Interfaces>
  <Routing>
    <Ospf>
      <RFC1583Compatible />
      <Areas>
        <Area id="0.0.0.0">
          <Networks>
            <Network>
              <IPAddress>130.10.63.0</IPAddress>
              <Wildcard>0.0.0.255</Wildcard>
            </Network>
            <Network>
              <IPAddress>130.10.62.0</IPAddress>
              <Wildcard>0.0.0.255</Wildcard>
            </Network>
          </Networks>
        </Area>
        <Area id="0.0.0.1">
          <Networks>
            <Network>
              <IPAddress>130.10.8.0</IPAddress>
              <Wildcard>0.0.0.255</Wildcard>
            </Network>
          </Networks>
        </Area>
      </Areas>
    </Ospf>
  </Routing>
</Router>
```

D.2 Konfigurační soubor pro RIP model

Uvedený model je použit u simulačního modelu, který využívá směrovací protokol RIP. Uvádíme jen konfiguraci pro směrovač R1.

```
<Router id="130.10.63.1"> <!-- R1 -->
  <Interfaces>
   <Interface name="eth0">
      <IPAddress>130.10.8.1</IPAddress>
```

```
      <Mask>255.255.255.0</Mask>
    </Interface>
    <Interface name="eth1">
      <IPAddress>130.10.62.1</IPAddress>
      <Mask>255.255.255.0</Mask>
    </Interface>
    <Interface name="eth2">
      <IPAddress>130.10.63.1</IPAddress>
      <Mask>255.255.255.0</Mask>
    </Interface>
  </Interfaces>
  <Routing>
    <Rip>
   <Network>130.10.0.0</Network>
   </Rip>
  </Routing>
</Router>
```

D.3 Konfigurační soubor pro redistribuční model

Uvedený model je použit u simulačního modelu, který využívá směrovací protokoly RIP a OSPF a vzájemnou redistribuci cest. Uvádíme jen konfiguraci pro směrovač R1.

```
<Router id="130.10.63.1"> <!-- R1 -->
  <Interfaces>
   <Interface name="eth0">
      <IPAddress>130.10.8.1</IPAddress>
      <Mask>255.255.255.0</Mask>
    </Interface>
    <Interface name="eth1">
      <IPAddress>130.10.62.1</IPAddress>
      <Mask>255.255.255.0</Mask>
      <OspfNetworkType>broadcast</OspfNetworkType>
      <OspfCost>1</OspfCost>
      <OspfPriority>1</OspfPriority>
      <OspfHelloInterval>10</OspfHelloInterval>
      <OspfDeadInterval>40</OspfDeadInterval>
      <OspfRetransmissionInterval>5</OspfRetransmissionInterval>
      <OspfInterfaceTransmissionDelay>1</OspfInterfaceTransmissionDelay>
      <OspfAuthenticationType>Null</OspfAuthenticationType>
      <OspfAuthenticationKey>0x00</OspfAuthenticationKey>
    </Interface>
    <Interface name="eth2">
      <IPAddress>130.10.63.1</IPAddress>
      <Mask>255.255.255.0</Mask>
      <OspfNetworkType>broadcast</OspfNetworkType>
      <OspfCost>1</OspfCost>
```

```
      <OspfPriority>1</OspfPriority>
      <OspfHelloInterval>10</OspfHelloInterval>
      <OspfDeadInterval>40</OspfDeadInterval>
      <OspfRetransmissionInterval>5</OspfRetransmissionInterval>
      <OspfInterfaceTransmissionDelay>1</OspfInterfaceTransmissionDelay>
      <OspfAuthenticationType>Null</OspfAuthenticationType>
      <OspfAuthenticationKey>0x00</OspfAuthenticationKey>
    </Interface>
  </Interfaces>
  <Routing>
    <Ospf>
      <RFC1583Compatible />
      <Areas>
        <Area id="0.0.0.0">
          <Networks>
            <Network>
              <IPAddress>130.10.62.0</IPAddress>
              <Wildcard>0.0.0.255</Wildcard>
            </Network>
            <Network>
              <IPAddress>130.10.63.0</IPAddress>
              <Wildcard>0.0.0.255</Wildcard>
            </Network>
          </Networks>
        </Area>
      </Areas>
    </Ospf>
    <Rip>
   <Network>130.10.0.0</Network>
       <Passive-interface>eth1</Passive-interface>
       <Passive-interface>eth2</Passive-interface>
       <Redistribute>
          <Protocol>ospf</Protocol>
          <Metric>4</Metric>
       </Redistribute>
   </Rip>
  </Routing>
</Router>
```