VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ
ÚSTAV TELEKOMUNIKACÍ

FACULTY OF ELECTRICAL ENGINEERING AND COMMUNICATION
DEPARTMENT OF TELECOMMUNICATIONS

NÁVRH SOFTWARU SLOUŽÍCÍHO K MAPOVÁNÍ TOPOLOGIE SÍTĚ

DIPLOMOVÁ PRÁCE
MASTER'S THESIS

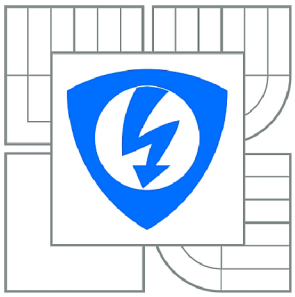
AUTOR PRÁCE
AUTHOR

Bc. DOMINIK JANURA

BRNO 2014

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ

ÚSTAV TELEKOMUNIKACÍ

FACULTY OF ELECTRICAL ENGINEERING AND COMMUNICATION

DEPARTMENT OF TELECOMMUNICATIONS

NÁVRH SOFTWARU SLOUŽÍCÍHO K MAPOVÁNÍ TOPOLOGIE SÍTĚ

SOFTWARE DESIGN FOR MAPPING OF THE NETWORK TOPOLOGY

DIPLOMOVÁ PRÁCE

MASTER'S THESIS

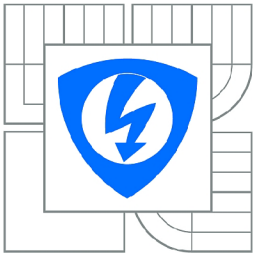
AUTOR PRÁCE — Bc. DOMINIK JANURA

AUTHOR

VEDOUCÍ PRÁCE — Ing. JIŘÍ SOBEK

SUPERVISOR

BRNO 2014



VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

Fakulta elektrotechniky a komunikačních technologií

Ústav telekomunikací

Diplomová práce

magisterský navazující studijní obor
Telekomunikační a informační technika

Student:	Bc. Dominik Janura	***ID:***	115192
Ročník:	2	***Akademický rok:***	2013/2014

NÁZEV TÉMATU:

Návrh softwaru sloužícího k mapování topologie sítě

POKYNY PRO VYPRACOVÁNÍ:

Prostudujte a následně zpracujte problematiku mapování topologie počítačových sítí složené z Cisco zařízení. Načrtněte a porovnejte možné způsoby řešení. Na základě těchto poznatků vytvořte software, který topologii sítě vykreslí. Získané výsledky vhodně prezentujte.

DOPORUČENÁ LITERATURA:

[1] ODOM, W. CCNP Route 642-902 official certification guide. Indianapolis: Cisco Press, c2010, xxxiv, 730 s. ISBN 978-1-58720-253-7.
[2] HUCABY, D. CCNP SWITCH 642-813 official
certification guide. Indianapolis: Cisco Press, c2010, xxvii, 459 s. ISBN 978-1-58720-243-8.
[3] KABELOVÁ, A., DOSTÁLEK, L. Velký průvodce protokoly TCP/IP a systémem DNS. 5., aktualiz. vyd. Brno: Computer Press, 2008, 488 s. ISBN 978-80-251-2236-5.

Termín zadání:	10.2.2014	***Termín odevzdání:***	30.5.2014

Vedoucí práce: Ing. Jiří Sobek
Konzultanti diplomové práce:

prof. Ing. Kamil Vrba, CSc.
Předseda oborové rady

UPOZORNĚNÍ:

ABSTRAKT

Táto práca sa zaoberá problematikou mapovania topológie sietí, pričom sieť je zložená z Cisco zariadení. Zahŕňa teoretické poznatky v tejto oblasti a bližšie popisuje možné postupy riešenia tohoto problému. Vysvetľuje postup pri návrhu a implementácii takéhoto softwaru pre mapovanie sietí zvolenými prostriedkami. Správna funkčnosť vytvorenej aplikácie je overená v prostredí virtuálnej siete.

KĽÚČOVÉ SLOVÁ

mapa, sieť, topológia, prepínač, smerovač, Cisco, Java

ABSTRACT

This thesis addresses the issues of network topology mapping, where the network consists of multiple Cisco devices. It includes theoretical knowledge in this field and closely describes possible solutions to this problem. It explains the process of design and implementation of this kind of network mapping software by chosen means. Correct function of the created application is tested on a virtual network.

KEYWORDS

map, network, topology, switch, router, Cisco, Java

JANURA, Dominik *Návrh softwaru slúžiaceho k mapovaniu topológie siete*: diplomová práca. Brno: Vysoké učení technické v Brně, Fakulta elektrotechniky a komunikačních technologií, Ústav telekomunikací, 2014. 61 s. Vedúci práce bol Ing. Jiří Sobek

PREHLÁSENIE

Prehlasujem, že som svoju diplomovoú prácu na tému „Návrh softwaru slúžiaceho k mapovaniu topológie siete“ vypracoval samostatne pod vedením vedúceho diplomovej práce, využitím odbornej literatúry a ďalších informačných zdrojov, ktoré sú všetky citované v práci a uvedené v zozname literatúry na konci práce.

Ako autor uvedenej diplomovej práce ďalej prehlasujem, že v súvislosti s vytvorením tejto diplomovej práce som neporušil autorské práva tretích osôb, najmä som nezasiahol nedovoleným spôsobom do cudzích autorských práv osobnostných a/nebo majetkových a som si plne vedomý následkov porušenia ustanovenia § 11 a nasledujúcich autorského zákona č. 121/2000 Sb., o právu autorském, o právoch súvisejúcich s právom autorským a o zmeně niektorých zákonov (autorský zákon), vo znení neskorších predpisov, vrátane možných trestnoprávnych dôsledkov vyplývajúcich z ustanovenia časti druhé, hlavy VI. diel 4 Trestného zákoníka č. 40/2009 Sb.

Brno ....

(podpis autora)

POĎAKOVANIE

Rád by som poďakoval vedúcemu diplomovej práce pánovi Ing. Jiřímu Sobkovi za odborné vedenie, konzultácie, trpezlivosť a podnetné návrhy k práci.

Brno ....

(podpis autora)

Faculty of Electrical Engineering
and Communication
Brno University of Technology
Purkynova 118, CZ-61200 Brno
Czech Republic
http://www.six.feec.vutbr.cz

POĎAKOVANIE

Výskum popísaný v tejto diplomovej práci bol realizovaný v laboratóriach podporených z projektu SIX; registračné číslo CZ.1.05/2.1.00/03.0072, operačný program Výzkum a vývoj pro inovace.

Brno ....

(podpis autora)

OBSAH

ZOZNAM OBRÁZKOV

ZOZNAM TABULIEK

ÚVOD

V súčasnej dobe je možné sledovať neustály rozvoj používaných počítačových sietí. S rastúcou komplexnosťou siete sa ale zároveň zvyšujú nároky na monitorovanie a zabezpečenie nepretržitého chodu siete. Jedným z najdôležitejších poznatkov a podmienkou pre efektívnu správu a optimalizáciu siete je určite znalosť jej úplnej topológie.

Hlavným cieľom tejto práce bude vytvoriť jednoduchý, užívateľsky prívetivý software, použiteľný na mapovanie topológie počítačovej siete. Predpokladom je, že táto sieť je zložená z Cisco zariadení a výsledná mapa sa bude skladať len z týchto aktívnych zariadení. Jej súčasťou nebudú koncové zariadenia, akými sú napríklad užívateľské stanice, sieťové tlačiarne a podobne. Mapovanie má prebiehať bez akéhokoľvek aktívneho sondovania sieťových prvkov. Zameranie na siete pozostávajúce výhradne z Cisco zariadení však umožňuje využiť textové konfiguračné súbory týchto zariadení ako jediný a postačujúci zdroj informácií pre zostavenie výslednej mapy siete.

Teoretická časť práce bude venovaná princípom fungovania skúmaných dvoch typov zariadení, t.j. prepínačov a smerovačov. V tejto sekcii sú bližšie popísané niektoré dôležité technológie a protokoly, ktoré tieto zariadenia využívajú. Ďalej sú rozoberané už existujúce možné metódy pri objavovaní a zostavovaní mapy siete a najbežnejšie sady protokolov, ktoré tieto metódy využívajú.

Teoretickou základňou pre túto časť práce budú vybrané publikácie z oboru počítačových sietí. Jedná sa najmä o knihu Networking Bible, ktorej autorom je B. Sosinsky. Je to obsiahla zbierka teoretických aj praktických poznatkov o moderných sieťových technológiách a môže slúžiť ako podrobný manuál ľuďom, ktorý sa chcú s problematikou bližšie oboznámiť.

Keďže práca je zameraná na siete skladajúce sa z Cisco sieťových zariadení, ďalšími zdrojmi budú aj viaceré publikácie vydané priamo touto spoločnosťou. Množstvo poznatkov je čerpaných najmä z publikácie CCNP Route642-902 Official Certification Guide, ktorej autorom je W. Odom.

Zvyšná časť práce sa bude už postupne venovať podrobnému rozboru problému, teoretickému návrhu riešenia a jeho praktickej realizácii. Budú tu popisované všetky informácie, ktoré bude nevyhnutné získať k zostaveniu mapy siete.

Implementácia riešenia bude využívať objektovo orientované programovacie paradigma. Pre realizáciu návrhu bol zvolený programovací jazyk Java s využitím viacerých existujúcich knižníc z dôvodov bližšie spomínaných v práci. Súčasťou práce bude samozrejme vytvorený spustiteľný program, ktorý bude schopný podľa návrhu spracovať ako vstup konfiguračné súbory typu `running-config`, `interfaces` a `mac-address-table` zariadení tvoriacich sieť a v jednoduchom grafickom užívateľskom rozhraní zobraziť mapu topológie siete.

Časť práce riešenie zadaného problému bude písaná formou podobnou návodu aby bola čo možno najzrozumiteľnejšia aj pre čitateľa neznalého problematiky.

Vytvorená aplikácia bude otestovaná na jednoduchej virtuálnej sieti zodpovedajúcej všetkými podstatnými vlastnosťami sieti reálnej. Sieť bude vytvorená v prostredí programu Cisco Packet Tracer, čo je sieťový simulačný program od firmy Cisco. Podporuje väčšinu Cisco zariadení a technológií, navyše umožňuje okamžitú a jednoduchú kontrolu topológie siete pre porovnanie s výstupom vlastného navrhnutého programu.

1 TEORETICKÁ ČASŤ PRÁCE

1.1 Sieťové zariadenia

Pod pojmom sieťové zariadenie si je možné predstaviť všetky typy zariadení, ktoré sú zapojené do konkrétnej počítačovej siete a sú schopné vysielať a prijímať dáta. Sieťový hardware je možné rozdeliť od najjednoduchších po najzložitejšie z hľadiska najvyššej vrstvy referenčného modelu ISO/OSI, na ktorej dané zariadenie pracuje. V skúmanej sieti budú rozlišované predovšetkým dva druhy sieťových zariadení, a to prepínače a smerovače.

1.1.1 Prepínač

Jedná sa o aktívny sieťový spojovací prvok. Umožňuje segmentáciu siete na oddelené kolízne domény predstavované jednotlivými portami daného prepínaču. Oproti opakovačom či rozbočovačom rozširuje funkčnosť o linkovú vrstvu (L2 prepínač), pracuje špeciálne s jej podvrstvou MAC (*Media Access Control*). L2 prepínač v podstate zodpovedá bráne podľa štandardu IEEE 802.1D [2].

Prepínač má za úlohu prepínanie rámcov. K tomu využíva informácie uložené vo vlastnej prepínacej tabuľke. Jej jednotlivé záznamy tvoria v podstate dvojice fyzická (MAC) adresa a príslušný port, za ktorým sa zariadenie s touto adresou nachádza. Prepínacia tabuľka je vytváraná automatickým procesom bez zásahu užívateľa. Prepínač do nej postupne zaznamenáva nielen cieľové, ale aj zdrojové fyzické adresy získané zo záhlaví prichádzajúcich rámcov. Samotné prepínanie potom prebieha tak, že sa cieľová adresa rámcu porovná s prepínacou tabuľkou a rámec odošle na zodpovedajúci port prepínaču. Ak nezodpovedá žiadnemu záznamu v tabuľke, (prípadne sa jedná o adresu všesmerovú) je rámec odoslaný na všetky ostatné porty prepínaču.

Z hľadiska spracovania prichádzajúcich rámcov sú podľa [25, str. 193] rozlišované tieto typy prepínania rámcov:

- *Cut through* – zo záhlavia rámcu prečíta cieľovú fyzickú adresu a rámec odošle na príslušný port bez akejkoľvek kontroly na chyby.
- *Store and forward* – prichádzajúce rámce ukladá do zásobníku, po overení kontrolného súčtu ich odosiela ďalej.
- *Fragment free* – pred odoslaním kontroluje prvých 64 bajtov rámcu, následne dôjde k prepínaniu. Kontrola celého rámcu je prenechaná protokolom vyšších vrstiev.
- Adaptívne prepínanie – schopnosť prepínaču dynamicky voliť medzi viacerými spomínanými typmi prepínania rámcov.

1.1.2 Smerovač

Smerovač je aktívne sieťové zariadenie, ktoré prepája dve a viac rôznych sietí. Jeho úlohou je nájsť optimálnu cestu k cieľovej stanici pre dáta prijaté smerovačom. Pracuje na sieťovej vrstve modelu ISO/OSI [26, str. 34-36], a teda prenášanými dátovými jednotkami sú pakety. Smerovače navzájom oddeľujú jednotlivé siete a filtrujú tak všesmerový prenos daných podsietí.

Riadia sa vždy určitým smerovacím mechanizmom, najrozšírenejším druhom je distribuované smerovanie. Spočíva v tom, že každý zo smerovačov si udržuje vlastnú smerovaciu tabuľku na základe komunikácie s ostatnými smerovačmi použitím niektorého zo smerovacích protokolov.

Každý smerovač potrebuje určité minimálne množstvo informácií, aby mohol úspešne a efektívne smerovať prichádzajúce pakety:

1. Cieľová adresa - IP adresu cieľového hosťa nesie každý prenášaný paket vo svojom záhlaví. Podľa adresy je paket smerovaný do príslušnej siete.
2. Susedné priamo pripojené smerovače - so susednými smerovačmi si smerovač vymieňa informácie o okolitých pripojených sieťach.
3. Možné cesty do okolitých sietí - zoznam všetkých možných ciest do pripojených sietí, tvoria ho dvojice adresa siete – rozhranie smerovaču, za ktorým sa sieť nachádza.
4. Najvhodnejšia cesta do každej siete – vyberaná zo zoznamu všetkých možných ciest podľa určitých podmienok – metriky alebo ceny trasy, prípadne administratívnej vzdialenosti trasy.
5. Spôsob spravovania smerovacích údajov – údaje o smerovaní sú uchovávané v smerovacej tabuľke, tá je definovaná buďto staticky alebo dynamicky pomocou smerovacieho protokolu.

Smerovacia tabuľka

Každý smerovač si udržuje vlastnú smerovaciu tabuľku. Tá obsahuje záznamy potrebné k rozhodovaniu o smerovaní paketov (viď tab. 1.1). Jednotlivé záznamy sú zoradené podľa dĺžky masky od najmenších sietí (najdlhšia maska siete) po najväčšie (najkratšia maska siete). To umožňuje efektívnejšie smerovanie. Posledný záznam predstavuje východziu bránu smerovaču (*default gateway*), na ktorú sú smerované pakety nezodpovedajúce žiadnemu predošlému záznamu smerovacej tabuľky. Proces prechádzania záznamov pri samotnom smerovaní prebieha podľa [25, str. 195-200] nasledovne:

1. Smerovač spočíta logický súčin cieľovej adresy a masky siete z daného riadku.
2. Výsledok porovná s cieľovou sieťou daného riadku tabuľky.
3. Ak došlo k zhode, paket je odoslaný na rozhranie uvedené v tomto zázname. V opačnom prípade je skúmaný nasledujúci riadok tabuľky.
4. Po prejdení všetkých záznamov tabuľky bez zhody je cieľ vyhodnotený ako nedosiahnuteľný.

Tab. 1.1: Príklad jedného záznamu smerovacej tabuľky.

Cieľová sieť	Maska siete	Východzia brána	Rozhranie	Metrika
192.168.0.0	255.255.255.0	192.168.0.100	192.168.0.100	10

Existujú dva druhy smerovacích tabuliek, a to statické a dynamické.

- Statické – záznamy sú do smerovacej tabuľky vkladané ručne pri konfigurácii smerovaču. Smerovač do tejto tabuľky v priebehu času nijako nezasahuje.
- Dynamické – jednotlivé smerovače v sieti si medzi sebou vymieňajú informácie o topológii a stave siete, z týchto informácií nasledovne zostavujú vlastné smerovacie tabuľky. Výmena informácií je zaistená smerovacími protokolmi.

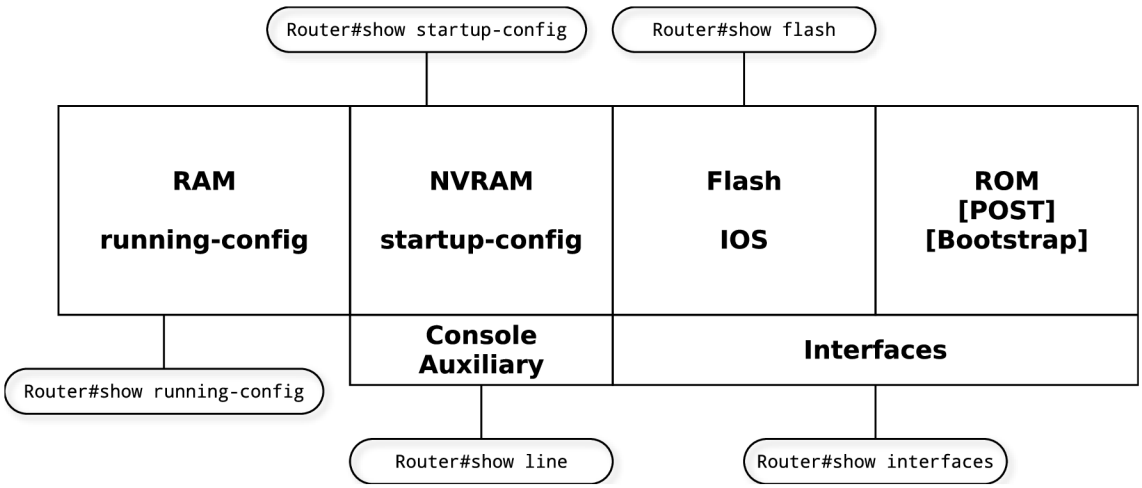
1.2 Cisco zariadenia a IOS

Smerovač je v podstate len veľmi špecializovaným typom počítaču, ako taký sa teda skladá z podobných súčastí (základná doska, procesor, operačná pamäť, rozhrania, atď.) a na svoj beh taktiež potrebuje operačný systém, ktorý bude obsluhovať všetky procesy spojené so smerovaním.

Hlavnými internými komponentmi Cisco smerovaču (viď obrázok 1.1 na strane 15) sú pamäte RAM (*Random Access Memory*), NVRAM (*Non-volatile Random Access Memory*), ROM (*Read-Only Memory*), pamäť flash a jeho jednotlivé rozhrania.

1. **RAM** – tiež označovaná ako DRAM (dynamická RAM) má tieto vlastnosti:
   - uchováva smerovacie tabuľky,
   - obsahuje vyrovnávaciu pamäť ARP,
   - poskytuje dočasné úložisko pre konfiguráciu `running-config`,
   - po vypnutí alebo reštarte stráca uchovávané dáta.
2. **NVRAM** – táto pamäť uchováva svoje dáta aj po vypnutí alebo reštarte zariadenia, je teda používaná ako úložisko pre konfiguráciu `startup-config`.

3. **ROM** – pamäť len pre čítanie, nie je možné na ňu dáta zapisovať. Udržuje sadu inštrukcií pre POST (*Power-on self test*) – diagnostiku pri spúšťaní zariadenia. Obsahuje software tzv. *bootstrap* – software spúšťaný po zapnutí zariadenia pred zavedením operačného systému IOS.
4. **Flash** – pamäť typu EEPROM (*Electrically Erasable Programmable Read-Only Memory*). Je to elektricky zmazateľná semipermanentná pamäť typu ROM. Základnou funkciou tejto pamäte je uchovávať jeden, prípadne viacero obrazov operačného systému IOS, umožňuje tak dostupnosť viacerých verzií systému zároveň. Umožňuje aktualizáciu softwaru bez nutnosti výmeny čipov procesoru [6].
5. **Interfaces** – rozhrania Cisco zariadenia vrátane sieťových rozhraní, konzolových portov a pod., umožňujú komunikáciu s ostatnými zariadeniami, sú buďto zabudované na základnej doske zariadenia alebo v podobe prídavných modulov.



Obr. 1.1: Základné interné komponenty Cisco smerovaču

1.2.1 Procesy smerovania

Cisco smerovače používajú tri druhy smerovania paketov:

1. *Process switching* – historicky prvý typ smerovania, vyžaduje pre každý jeden prichádzajúci paket prechádzať každým jedným záznamom smerovacej tabuľky a vyhľadávanie výstupného rozhrania. Takýto proces je pri obrovskom množstve záznamov, ktoré môže smerovač uchovávať, výpočtovo aj časovo veľmi náročný a neefektívny.

2. *Fast switching* – Proces rýchleho smerovania využíva vyrovnávaciu pamäť, v ktorej je uchovávaný určitý počet naposledy použitých destinácií, tieto potom nemusia byť znovu vyhľadávané v smerovacej tabuľke a smerovanie je tak urýchlené.
3. *Cisco express forwarding* – využíva FIB (*Forwarding Information Base*), ktorá je podobná smerovacej tabuľke, uchováva len adresu najbližšieho suseda pre každú trasu. Ďalej používa tabuľku susedov (*adjacencies*) s prepínacími informáciami druhej vrstvy, každý záznam je priradený k záznamu z FIB [9]. Tým je zabránené vytváraniu ARP požiadavkov pre každý vyhľadaný záznam z tabuľky. Tieto tabuľky sa aktualizujú pri zmene topológie siete [8, str. 342].

Spoločnosť Cisco Systems poskytuje veľké množstvo rôznych prepínačov, smerovačov a iných sieťových zariadení. Prakticky všetky smerovače a väčšina prepínačov triedy Cisco Catalyst ale majú jednu spoločnú vlastnosť, sú obsluhované pomocou operačného systému IOS (*Internetwork Operating System*). Prvá verzia systému IOS bola vytvorená W. Yaegerom v roku 1986 [8, str. 216]. Spoločný operačný systém pre rôzne typy zariadení je na jednú stranu veľkou výhodou z hľadiska univerzálnosti – jednou a tou istou sadou príkazov je možné ovládať veľké množstvo rôznych zariadení. Naopak nevýhodou takéhoto prístupu je komplexnosť operačného systému, ten je vždy rovnako rozsiahly bez ohľadu na rôzne využitia konkrétnych zariadení.

Pred vykonávaním samotnej konfigurácie, kontroly konfigurácie či vo všeobecnosti akejkoľvek komunikácie s Cisco zariadením je potrebné sa najprv na toto zariadenie pripojiť. To je možné uskutočniť rôznymi spôsobmi. Prvým spôsobom je fyzické pripojenie použitím konzolového portu alebo prídavného portu (*auxiliary port*). Oba porty používajú 8-pinový RJ-45 konektor [8, str. 216], rozdiel v nich je v tom, že prídavný port umožňuje navyše zadávanie príkazov pre modem. Cisco prepínače na rozdiel od smerovačov prídavný port nemajú. Pripojenie pomocou týchto portov môže ale nemusí byť chránené heslom v závislosti od konfigurácie zariadenia.

Ďalšou možnosťou pripojenia k Cisco zariadeniu je tzv. vzdialené pripojenie, a to použitím jedného z protokolov Telnet alebo SSH (*Secure Shell*). Protokol Telnet umožňuje užívateľovi pripojenie pomocou rovnomennej aplikácie. Na prenos dát využíva protokol TCP, pričom serverová časť štandardne používa port s číslom 23. Podstatnou nevýhodou použitia protokolu Telnet je absencia akéhokoľvek šifrovania prenášaných dát (vrátane prihlasovacích údajov), tie tak môžu byť počas prenosu získané a zneužité treťou stranou.

Z tohto dôvodu použitie protokolu Telnet postupne upadá a je nahradzované použitím protokolu SSH. Ten poskytuje obdobné služby ako protokol Telnet, ale navyše umožňuje:

- autentizáciu oboch účastníkov spojenia,
- transparentné šifrovanie prenášaných dát,
- zabezpečenie integrity prenášaných dát,
- voliteľnú bezstratovú kompresiu prenášaných dát [20].

SSH server používa štandardne TCP port číslo 22. Protokol využíva asymetrickú šifru, a teda dvojicu šifrovacích kľúčov, verejný kľúč pre šifrovanie a súkromný kľúč pre dešifrovanie správ.

Po pripojení k Cisco zariadeniu je ďalej užívateľovi umožnená jeho správa. Operačný systém IOS využíva pre správu komplexnú sadu príkazov tvoriacich konfiguračný jazyk systému. Pritom konfigurácie jednotlivých zariadení si je možné predstavovať ako textové súbory. Samostatné riadky takéhoto súboru tvoria príkazy ovplyvňujúce chovanie tohto zariadenia. Príkazy sú v súbore logicky štrukturované do sekcií parametrizujúcich napríklad sieťové rozhrania či procesy bežiace na zariadení (viď ukážku konfigurácie na strane 18).

Každé zariadenie uchováva dve hlavné kópie svojho konfiguračného súboru. Konfigurácia `running-config`, ktorú zariadenie uchováva a mení za behu, je uložená v pamäti RAM (*Random Access Memory*). Z toho vyplýva, že táto konfigurácia nie je perzistentná, vymazáva sa pri každom reštarte zariadenia.

Druhou kópiou konfiguračného súboru je súbor `startup-config`. Ten je uložený v pamäti NVRAM (*Non-volatile Random Access Memory*), čo je energeticky nezávislý druh pamäte, čiže dáta uchováva aj po reštarte zariadenia. Táto konfigurácia je teda perzistentná.

Výpis kódu 1.1: Ukážka konfigurácie zo súboru `running-config`

```
: Saved
 :
 XXX Version X.X(X)
 names
 !
 interface Ethernet0
  nameif test
  security-level 10
  ip address 10.10.88.50 255.255.255.254
 !
 enable password 8Ry2YjIyt7RRXU24 encrypted
 passwd 2KFQnbNIdI.2KYOU encrypted
 hostname XXX
```

```
 domain-name XXX.com
 boot system flash:/cdisk.bin
 ftp mode passive
: end
```

Zmenu konfigurácie v RAM je možné trvalo uložiť do NVRAM použitím nasledujúceho IOS príkazu.

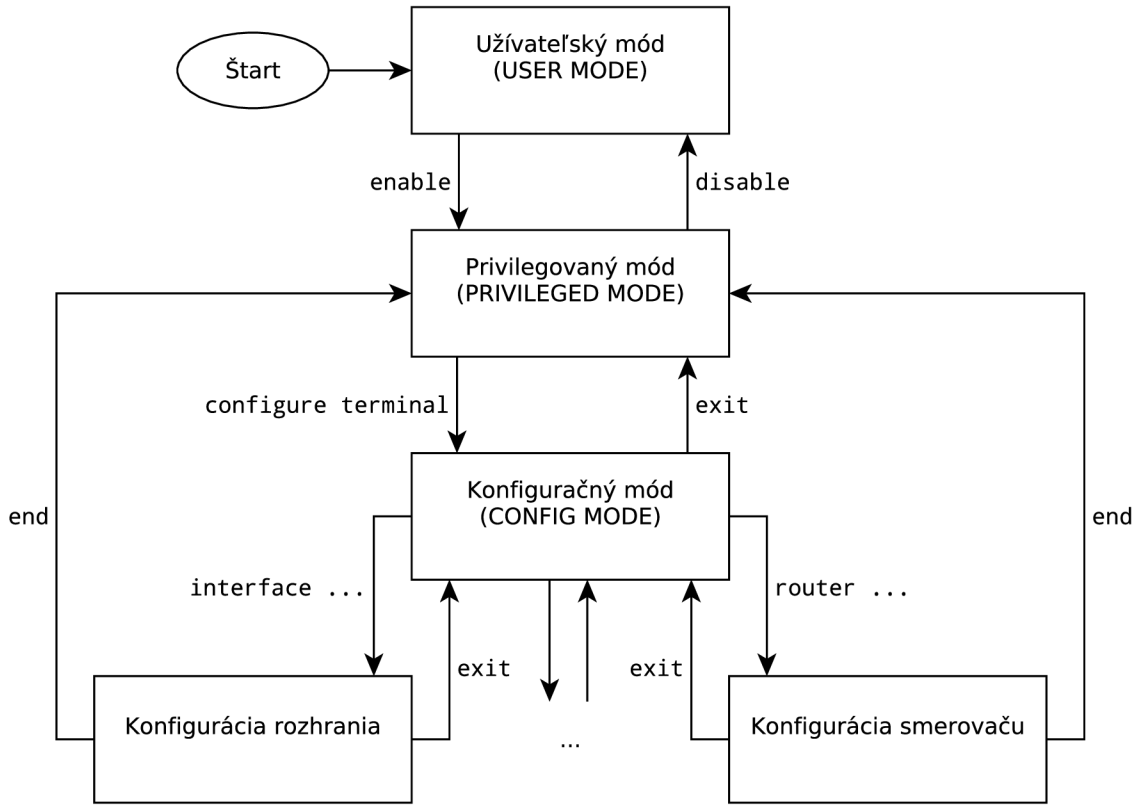
```
copy running-config startup-config
```

Nastavenia z „`running-config`“ je taktiež možné zálohovať použitím protokolu TFTP (*Trivial File Transfer Protocol*) v prípade, že je v sieti dostupný TFTP server.

```
copy running-config tftp
```

Práca s Cisco zariadením môže prebiehať v niekoľkých základných režimoch (viď obr. 1.2 na strane 19), ktoré sa odlišujú právami a možnosťami konfigurácie. Jedná sa o nasledujúce režimy.

1. **Užívateľský mód** – východzí režim po prihlásení na Cisco zariadenie, poskytuje veľmi obmedzené možnosti a informácie, spravidla výpis základných informácií o hardware a použitej verzii operačného systému IOS [8, str. 219].
2. **Privilegovaný mód** – umožňuje výpis všetkých dostupných informácií o zariadení a okrem toho spúšťanie niektorých príkazov (napr. `ping`). Príkazy sa však môže len vykonať, nesmú zasahovať do konfigurácie zariadenia. Do privilegovaného módu je možné prejsť z módu užívateľského príkazom `enable`. Prechod môže v závislosti na konfigurácii zariadenia vyžadovať autentizáciu užívateľa. Opačný prechod zabezpečuje obdobne príkaz `disable`.
3. **Konfiguračný mód** – tento režim umožňuje zadávanie príkazov ovplyvňujúcich priamo konfiguráciu zariadenia. Príkazy sa okamžite stávajú súčasťou konfiguračného súboru `running-config` a zariadenie taktiež priamo podľa nich mení svoj stav. Sada príkazov je samozrejme odlišná od privilegovaného módu (napr. `interface`, `line`, `router`, `ip access-list`, atď.) [8, str. 221]. Do globálneho konfiguračného režimu je možné prejsť príkazom `configure terminal`. Ďalej je možné prechádzať do konfigurácie jednotlivých sekcií, napríklad konkrétneho rozhrania
   `interface Serial0/1/0`,
   ktoré má opäť vlastnú sadu príkazov.



Obr. 1.2: Základné módy systému IOS a príkazy na prechod medzi nimi

1.3 Objavovanie a mapovanie topológie sietí

Objavovanie siete je procesom získavania informácií o tom, ako sú jednotlivé zariadenia v sieti usporiadané a prepojené. Obyčajne sa jedná o postup, pri ktorom jedna zo staníc v sieti objavuje ostatné stanice a zariadenia. Mapovanie topológie siete je potom proces, pri ktorom sú graficky vyobrazené objavené sieťové prvky a spojenia. Mapovaniu topológie teda musí vždy predchádzať aj proces objavovania siete. Existuje viacero základných metód objavovania siete, pričom tieto metódy sú spravidla nezávislé od použitého protokolu. Využitie niektorých špecifických protokolov je však pri procesoch objavovania siete výhodnejšie.

1.3.1 Základné metódy objavovania sietí

Využitie všesmerových správ

Medzi najjednoduchšie metódy objavovania siete patrí oznámenie pomocou všesmerovej správy určitého protokolu. V takomto prípade dané zariadenie v sieti po inicializácii svojo sieťového rozhrania najprv nájde zariadenia podporujúce daný protokol pomocou IP broadcastu, následne odošle všesmerovú správu daného protokolu. Táto správa oznamuje, že zariadenie je aktívne a obsahuje adresu jeho rozhrania [5].

Oznámenia pomocou všesmerových správ sú však vhodné skôr pre získavanie informácií z len jedného, prípadne niekoľko málo zariadení v sieti, akým môže byť napríklad DHCP server. Inak by mohlo dochádzať ku zbytočnému zahlcovaniu liniek.

Nevýhodou tejto metódy je použitie IP broadcastu, ktorý jednak nie je podporovaný v IPv6 ale taktiež nedosahuje do smerovačom oddelených segmentov siete. Metóda je teda aplikovateľná len v malých IPv4 sieťach.

Trasovanie spojenia k jednotlivým uzlom siete

Táto metóda využíva správy protokolu ICMP a obdobne ICMPv6 v prípade IPv6. Konkrétne sa jedná o správy ICMP *Echo Request* (požiadavok) a ICMP *Echo Reply* (odpoveď).

Samotné trasovanie, ako uvádza [24], využíva dobu života paketu TTL (*Time To Live*). Je to 8bitové pole v záhlaví IP datagramu, ktoré obmedzuje maximálnu dobu existencie datagramu, a tak pomáha zabrániť zahlcovaniu siete. Pri vytvorení datagramu je nastavená na východziu hodnotu a je znížená pri každom preposlaní uzlom siete. Ak dosiahne hodnotu 0 pred príchodom datagramu do cieľovej stanice, je datagram zahodený a pôvodný odosielateľ je informovaný o nedosažiteľnosti cieľa správou typu ICMP *Time Exceeded*.

Zo získaných informácií je možné zostaviť kompletnú trasu datagramu jednotlivými uzlami a po skombinovaní viacerých takto skúmaných trás aj mapu siete. To však vyžaduje trasovanie ku každému potencionálne aktívnemu uzlu siete, čo zásadne limituje rozsah skúmanej siete.

Výhodou metódy je teda nezávislosť na protokoloch iných než IP, naopak nevýhodami sú časová náročnosť metódy, veľké množstvo vytváraných spojení a praktická nepoužiteľnosť v IPv6 sieťach.

K trasovaniu spojenia je možné využiť program `traceroute` [15].

Metóda postupného dotazovania

Metóda postupného dotazovania vyžaduje ako vstupnú informáciu adresu aspoň jedného susedného uzlu, podporujúceho vybraný protokol správy. Na tento uzol bude iniciované prvé spojenie. Informácie získané z odpovede sú nasledovne priamo použité pri vytvorení ďalšieho spojenia. Postupným opakovaním takéhoto postupu dochádza k prechádzaniu veľkej časti siete, v ideálnom prípade celej siete. Jedná sa o pomalší proces objavovania siete, a to z dôvodu nutnosti čakať na odpoveď jednotlivých uzlov siete [3].

To je ale kompenzované tým, že vytvárame spojenia len na adresy získané z odpovedí uzlov siete, čím sa počet vytvorených spojení a tým aj doba spracovania výrazne minimalizuje.

Výhodou je aj to, že takýto postup nie je limitovaný na lokálnu sieť, vďaka priamej adresácii môže dotazovanie pokračovať aj do iných podsietí.

Hlavnou nevýhodou tejto metódy je možnosť vzniku „slepých“ vetiev siete, kedy hraničný uzol nepodporuje použitý protokol správy, a tak môže zostať značná časť siete neobjavená.

Využitie smerovacieho protokolu

K zostaveniu mapy celej siete je možné využiť aj smerovacie protokoly typu *link-state*. Mapovanie u nich podľa [11] prebieha nasledovne:

1. Každý uzol vytvára vlastný zoznam pripojených liniek a susedných zariadení.
2. Uzly rozosielajú svojim susedom oznámenia, ktoré obsahujú tento zoznamom. Po prijatí takéhoto oznámenia ho uzol rozošle na všetky susedné uzly okrem toho, z ktorého oznámenie prijal. Nekontrolovateľnému šíreniu je zabránené tým, že každý uzol rozosiela jedno konkrétne oznámenie len raz, ak ho teda obdrží znovu, už ho ďalej neodosiela.
3. Zo zoznamu uzlov a spojení je vytvorená mapa siete.

Jedným zo smerovacích protokolov je aj protokol OSPF. Viac informácií je uvedených v sekcii 1.3.2 na strane 24.

Nevýhodou tejto metódy je skutočnosť, že smerovacie protokoly pracujú spravidla na sieťovej vrstve modelu ISO/OSI, a zariadenia nižších vrstiev (napr. prepínače) preto do mapy siete nie sú zahrnuté.

1.3.2 Protokoly vhodné pre objavovanie sietí

Internet Protocol

Internet Protocol je základným protokolom v dnešných sieťach poskytujúcim spojenie typu bod-bod. Existujú dve jeho verzie, IPv4, ktorý je popísaný v RFC 791 [12] z roku 1981 a jeho nástupcu IPv6, ktorého nasadzovanie sa postupne rozširuje. Pracuje na sieťovej vrstve modelu ISO/OSI a protokolom vyšších vrstiev poskytuje datagramovú službu. K získaniu adries nižších vrstiev zase využíva protokol pre preklad adries ARP (*Address Resolution Protocol*).

Adresácia sieťových zariadení je zabezpečená IP adresami o dĺžke 32 bitov. Väčšinou sa zapisujú v tvare štyroch oktetov v desiatkovej sústave oddelených bodkami. Adresný priestor môže byť rozdelený do rôzne veľkých podmnožín s využitím masky siete. Tá adresu rozdeľuje na adresu siete a adresu sieťového rozhrania. V binárnom zápise má na miestach adresy siete samé jedničky a na miestach adresy sieťového rozhrania samé nuly. Techniky podsieťovania a napríklad prekladu adries NAT (*Network Address Translation*) umožňujú rozšíriť adresný IPv4 priestor. Technológia NAT je špecifikovaná štandardom RFC 1631 [16].

V IPv6 priestor adresovaný 128-bitovými adresami. Ich štandardný zápis je v hexadecimálnom tvare po štvoriciach oddelených dvojbodkou. Veľkosť záhlavia paketu bola oproti IPv4 zredukovaná, nová verzia protokolu ICMPv6 kombinuje funkcie ICMP (*Internet Control Message Protocol*), IGMP (*Internet Group Membership Protocol*), a ARP [21].

Protokol IP patrí medzi nespojované typy protokolov. Znamená to, že pred komunikáciou nie je zostavená pevná cesta, datagramy sú odosielané nezávisle. Protokol pozná len počiatočný a koncový bod komunikácie. Z tohto dôvodu je relatívne nezávislý na type sietí. Pracuje na sieťach s prepínaním paketov s technológiami ako sú Ethernet, ATM, FDDI či bezdrôtové siete IEEE 802.11 [25, str. 476].

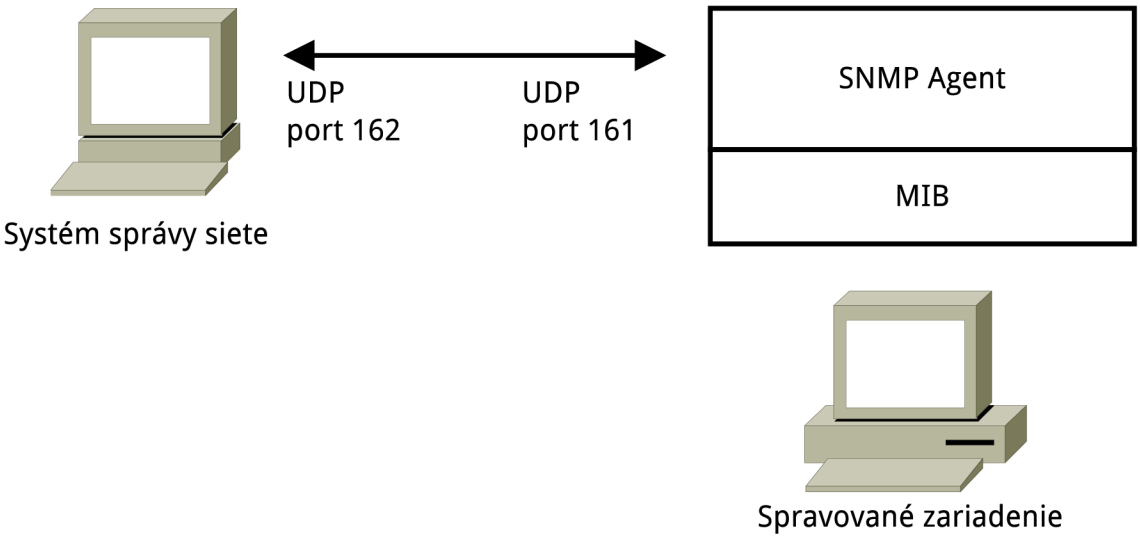
Simple Network Management Protocol

Protokol SNMP patrí k protokolom aplikačnej vrstvy modelu ISO/OSI. Je súčasťou sady internetových protokolov TCP/IP. Slúži k zberu informácií potrebných k správe siete a ich následnému vyhodnocovaniu. Celý systém SNMP je možné rozdeliť na tri časti:

- Spravovaný objekt – sieťové zariadenia akými sú napríklad sieťové karty, prepínače či smerovače.
- Agent – softwarový modul bežiaci na spravovanom objekte, poskytuje informácie zozbierané z daného objektu a sieťovej komunikácie SNMP požiadavkom.
- Systém správy siete NMS (*Network Management System*) – software bežiaci na tzv. správcovi siete.

V SNMP komunikácii (príklad na obr. 1.3) teda vystupujú dve strany, strana monitorovacia – správca siete a strana monitorovaná – spravovaný objekt. Tieto časti môžu vystupovať buďto oddelene na rôznych fyzických zariadeniach, alebo aj na jednom fyzickom stroji. [25, str. 73]

Management Information Base je databázovou štruktúrou, ktorú protokol SNMP využíva k správe sieťových entít. Jedná sa o hierarchickú štruktúru a je podrobne popísaná v dokumentácii RFC 1155 [13] a RFC 1213 [14].



Obr. 1.3: Systém SNMP

Link Layer Discovery Protocol a Cisco Discovery Protocol

Link Layer Discovery Protocol je protokolom linkovej vrstvy a je súčasťou protokolovej sady TCP/IP. Sieťové zariadenia ho používajú k propagácii rôznych informácií, ako sú vlastná identita, schopnosti zariadenia či najbližšie susedné zariadenia, v rámci lokálnej siete podľa štandardu IEEE 802. Narozdiel od CDP je LLDP štandardizovaným protokolom popísaným v dokumentácii IEEE 802.1AB [1].

Informácie, ktoré sú týmto protokolom distribuované, sú ukladané v štandardnej MIB, a tak k nim môžu pristupovať aj rôzne protokoly na správu sietí (napr. protokol SNMP).

Cisco Discovery Protocol je proprietárnym protokolom spoločnosti Cisco Systems a je priamou alternatívou k protokolu LLDP. Obdobne sa využíva na zdieľanie informácii o všetkých priamo pripojených Cisco zariadeniach. CPD správy sú vysielané na špeciálnu multicastovú adresu `01-00-0c-cc-cc-cc`. Tieto správy môže prijať každé zariadenie podporujúce protokol CDP a obsahujú informácie, ako napríklad adresa odosielateľa aj príjemcu, typ a názov zariadenia, hardwarovú platformu zariadenia či použitý port (podľa [4]). Tieto informácie sú taktiež ukladané do MIB. Jeho výhodou oproti protokolu LLDP je možnosť zahrnúť do správ smerovacie informácie, a tak ho využiť namiesto dynamických smerovacích protokolov, predovšetkým v menších a jednoduchších sieťach.

Open Shortest Path First

Smerovací protokol je protokol, ktorý medzi smerovačmi v sieti prenáša informácie o výbere najideálnejšej trasy medzi dvomi bodmi. OSPF je v súčasnosti jedným z najrozšírenejších typov smerovacích protokolov. Používa smerovací algoritmus typu *link-state*, pričom každý uzol v sieti zostavuje vlastnú mapu siete vo forme grafu. Pre IPv4 je definovaný ako OSPF verzie 2 v RFC 2328 [18] a pre IPv6 ako OSPF verzie 3 v RFC 5340 [22].

V pamäti smerovaču vytvára kompletnú topologickú databázu celej siete. Pre nájdenie ideálnej trasy sieťou využíva Dijkstrov algoritmus najkratšej cesty SPF (*Shortest Path First*). Patrí medzi *classless* protokoly, v informáciach o dosažiteľnosti cieľa posiela masku podsiete. Podporuje VLSM – variabilnú dĺžku masky podsiete. Veľké siete rozdeľuje do oblastí (*areas*), vďaka čomu sa znižuje výpočtová náročnosť algoritmu SPF a zmenšujú smerovacie tabuľky a taktiež prenášané správy LSU.

Priľahlé smerovače si navzájom vymieňajú informácie pomocou správ LSU (*Link State Update*), ktoré obsahujú oznámenia o stave linky LSA (*Link State Advertisement*). Tie sa ukladajú do *link-state* databázy, vďaka ktorej sa predchádza smyčkám v sieti. Správy LSU sa posielajú len v prípade ak je ich potreba, nie periodicky. Pri výpočte samotnému algoritmu SPF sa používa metrika linky, počíta sa z rýchlosti

linky ako podiel $1/r$, kde r je prenosová rýchlosť linky v bitoch za sekundu (viď tab. 1.2).

Tab. 1.2: Metriky bežne používaných typov liniek

Typ linky	Prenosová rýchlosť [b/s]	cena
Fast Ethernet	10^8	1
Ethernet	10^7	10
E1	$2,048 \cdot 10^6$	48
T1	$1,544 \cdot 10^6$	64
128 kbps	$1,28 \cdot 10^5$	781
64 kbps	$6,4 \cdot 10^4$	1562
56 kbps	$5,6 \cdot 10^4$	1785

Protokol OSPF podporuje autentizáciu v čistom texte alebo šifrovanú pomocou MD5. Jeho výhodou je zaručená a rýchla konvergencia pri hľadaní cesty.

2 NÁVRH VLASTNÉHO RIEŠENIA

Cieľom tohto vlastného návrhu je vytvoriť aplikáciu vhodnú pre použitie bežným užívateľom a bez nutnosti aktívneho sondovania siete. Všetky informácie potrebné pre zostavenie a vykreslenie mapy siete budú získané len z textových konfiguračných súborov Cisco zariadení, z ktorých je sieť zložená. Z toho vyplýva, že mapa nebude obsahovať koncové uzly siete, akými sú napríklad osobné počítače či sieťové tlačiarne, ale iba prenosové uzly, a to najbežnejšie typy zariadení – Cisco prepínače a smerovače. Navrhnutý program bude podporovať adresáciu v sieti protokolom IPv4. Podpora protokolu IPv6 nie je v rozsahu cieľov tejto práce.

Najjednoduchším prístupom na vyriešenie tohto problému by bolo priame využitie protokolu CDP, vyvinutým priamo firmou Cisco Systems (viď. 1.3.2). Údaje z tohto protokolu sú dostupné po zadaní príkazu `show cdp neighbors` do príkazového riadku podporovaného zariadenia.

Výpis kódu 2.1: Výstup príkazu `show cdp neighbors`

```
center#show cdp neighbors
Capability Codes: R - Router, T - Trans Bridge, B - Source Route
   Bridge
                 S~- Switch, H - Host, I~- IGMP, r - Repeater, P -
    Phone
Device ID    Local Intrfce   Holdtme    Capability   Platform
   Port ID
office3      Gig 0/1          163            S~2960        Fas 0/1
office1      Gig 0/2          163            S~2960        Fas 0/4
west         Gig 0/0          163            R       C2900
   Gig 0/0
north        Ser 0/0/0        171            R       C2900
   Ser 0/0/0
```

Výstup príkazu obsahuje všetky nevyhnutne potrebné údaje pre zostavenie mapy siete. Sú to hostiteľské názvy pripojených zariadení (pod položkou Device ID), ich typ (položka Capability) a dvojica lokálne – vzdialené rozhranie (pod položkami Local Intrfce, resp. Port ID), ktorá tvorí samotné spojenie.

Pre kompletnú mapu siete by tak ako vstup stačila dvojica súborov priradená každému uzlu v siete, a to súbory `running-config` a `cdp neighbors`.

Využitie protokolu CDP ale nie je vhodným univerzálnym riešením. Aj keď Cisco zariadenia majú podporu tohto protokolu v pôvodnej konfiguráciu zapnutú, z dôvodu rôznych bezpečnostných rizík býva často táto podpora administrátormi siete vypnutá. [23]

Najbežnejšou formou útokov zneužívajúcich protokol CDP je útok typu CDP spoofing, ktorý je špecifickým druhom DoS (*Denial of Service*) útokov.

Útok prebieha nasledujúcim spôsobom: útočník posiela veľké množstvo CDP správ s podvrhnutou zdrojovou fyzickou adresou na multicastovú adresu `01-00-0c-cc-cc-cc`. Zariadenie, ktoré je cieľom útoku sa následne snaží pridať všetky tieto informácie do vlastnej CDP tabuľky. Ak je objem prichádzajúcich rámcov dostatočne veľký, zariadenie ich prestáva zvládať spracovať a dochádza tak k zahlteniu.

Z týchto dôvodov sa táto práca nebude využitím protokolu CDP zaoberať. K mapovaniu siete budú použité len informácie zo základných príkazov Cisco zariadení, akými sú `show running-config`, `show interfaces` a `show mac-addresstable`.

K zostaveniu mapy siete bude potrebných niekoľko podstatných informácií. Prvým z nich je *hostname* (názov hostiteľa), ktorým budú identifikované jednotlivé uzly vytvorenej mapy. Tento údaj sa v konfiguračnom súbore Cisco zariadení nachádza pod rovnomennou položkou `hostname`, ako vidno na príklade úryvku z výpisu príkazu `show running-config`:

Výpis kódu 2.2: Časť výstupu príkazu `show running-config`

```
hostname Transit2
!
interface GigabitEthernet0/0
 ip address 8.20.20.1 255.255.255.252
 duplex auto
 speed auto
!
interface GigabitEthernet0/1
 no ip address
 duplex auto
 speed auto
 shutdown
!
```

Ďalšou hľadanou informáciou bude typ daného zariadenia, a teda buďto prepínač, alebo smerovač. Konkrétny údaj o type zariadenia sa však v konfiguračnom súbore priamo nenachádza. Je teda nutné vychádzať z predpokladu, že Cisco prepínače nemajú u žiadnych svojich fyzických rozhraní definovanú IP adresu. Ak sa teda po prečítaní konfigurácie týchto rozhraní nebola nájdená žiadna IP adresa, jedná sa o prepínač, v opačnom prípade zase o smerovač.

Tým sú skompletizované údaje o jednotlivých uzloch, chýbajú však ešte dôležitejšie informácie, a to tie o prepojení a teda aj rozostavení týchto uzlov. Tu sa postup použiteľný pre prepínače a pre smerovače začína odlišovať.

U smerovačov je potrebné vytvoriť zoznam aktívnych sieťových rozhraní, a teda rozhraní, ktorým je pridelená IP adresa. U každého takéhoto aktívneho rozhrania je navyše potrebné zistiť jeho názov, IP adresu, masku podsiete, do ktorej je toto rozhranie pripojené a adresu tejto podsiete. Tá v konfigurácii uvedená nie je, a to

z dôvodu, že je ju jednoduché spočítať pomocou logického súčinu IP adresy a masky v binárnom zápise (viď tab. 2.1). Príklad konfigurácie aktívneho rozhrania smerovaču:

```
interface Ethernet0
ip address 10.10.88.50 255.255.255.252
```

Tab. 2.1: Príklad výpočtu adresy podsiete

Položka	Binárna IP	Decimálna IP
IP adresa	00001010.00001010.01011000.0011**0010**	10.10.88.50
Maska	11111111.11111111.11111111.1111**0000**	255.255.255.240
operácia AND		
Sieť	00001010.00001010.01011000.0011**0000**	10.10.88.48

Po získaní spomenutých údajov pre každý smerovač v sieti už potom nie je problém zostaviť mapu siete. Stačí vytvoriť spojenia medzi dvojicami smerovačov, ktoré majú aktívne rozhrania patriace do spoločnej podsiete.

V prípade prepínačov sa postup komplikuje tým, že pracujú len na linkovej vrstve modelu ISO/OSI. Konfigurácia ich fyzických rozhraní teda nemôže obsahovať adresy protokolu IP, ale len fyzické MAC adresy. Namiesto dvojice rozhranie – IP adresa je potom hľadanou dvojicou rozhranie – MAC adresa. U každého smerovaču bude teda potrebné uchovávať fyzické adresy všetkých aktívnych rozhraní. Fyzickú adresu každého z jednotlivých rozhraní zariadenia je možné zistiť príkazom `show interfaces` a na strane prepínaču sa tabuľka MAC adries vypíše po zadaní príkazu `show mac-address-table`. V nej je uložený údaj o fyzickej adrese východzej brány, na ktorú je konkrétne rozhranie pripojené.

Okrem výpisu `running-config` budú teda potrebné ako vstup do aplikácie na mapovanie siete aj výpisy týchto dvoch príkazov.

Časť výstupu príkazu `show interfaces` u Cisco smerovaču:

Výpis kódu 2.3: Časť výstupu príkazu `show interfaces`

```
Router#show interfaces
GigabitEthernet0/0 is up, line protocol is up (connected)
Hardware is CN Gigabit Ethernet, address is 000a.f377.4d01 (bia 000
   a.f377.4d01)
```

a odpovedajúci výstup príkazu `show mac-address-table` u prepínaču:

Výpis kódu 2.4: Výstup príkazu `show mac-address-table`

```
Switch#show mac-address-table
          Mac Address Table
-------------------------------------------

Vlan    Mac Address       Type        Ports
----    -----------       --------    -----

   1    000a.4170.1e01    DYNAMIC     Fa0/1
   1    0090.2b05.7402    DYNAMIC     Fa0/1
```

Je možné vidieť, že tabuľka MAC adries tohto prepínaču obsahuje adresu jedného z rozhraní smerovaču. Po nájdení takejto odpovedajúcej dvojice záznamov o fyzickej adrese potom do mapy siete stačí zakresliť spojenie medzi príslušnými rozhraniami týchto dvoch zariadení.

2.1 Problém privátnych sietí

Pri vzájomnom mapovaní rozhraní smerovačov podľa pridelenej adresy siete je možné naraziť na problém pri použití privátnych rozsahov adries (viď tab. 2.2). Privátne adresy (popísané podľa štandardov daných RFC 1918 [17] a RFC 4193 [19]) sú často využívané v lokálnych sieťach, kde verejné adresy nie sú potrebné alebo, v častejšom prípade, nie sú dostupné.

Adresy z privátneho rozsahu sa v kompletnej sieti môžu opakovať neobmedzene veľa krát, z toho vyplýva, že nie sú globálne smerovatelné. A teda ani základný princíp mapovania v rámci navrhovaného programu (vytvárať spojenia medzi dvojicami rozhraní na základe rovnakej adresy siete) nebude v privátnom rozsahu fungovať.

Keďže ale v sieťach väčšieho rozsahu, u ktorých má použitie tohto programu najväčší zmysel, bývajú dostupné adresy verejné a naopak u menších lokálnych sietí potreba automatického mapovania odpadá, práca bude založená na predpoklade, že minimálne medzi dvojicami smerovačov budú použité adresy verejné.

V podsieťach medzi dvojicou smerovač – prepínač budú pre účely mapovania použité fyzické adresy rozhraní, v tomto prípade teda na adrese siete nezáleží a môže byť z ľubovolného rozsahu.

Tab. 2.2: Privátny priestor IPv4 adries

Označenie RFC 1918	Rozsah adries	Počet adries	Maska
24-bitový blok	10.0.0.0 – 10.255.255.255	16.777.216	10.0.0.0/8
20-bitový blok	172.16.0.0 – 172.31.255.255	1.048.576	172.16.0.0/12
16-bitový blok	192.168.0.0 – 192.168.255.255	65.536	192.168.0.0/16

2.2 Návrh štruktúry pre uchovávanie dát

Pri návrhu softwaru je dôležité všetky dáta, s ktorými pracuje uchovávať vo vhodne organizovanej podobe, ktorá bude dobre dostupná a čitateľná ako pre stroj, tak aj pre človeka. Dátový model bol navrhnutý a prispôsobený použitiu objektovo orientovaného programovania.

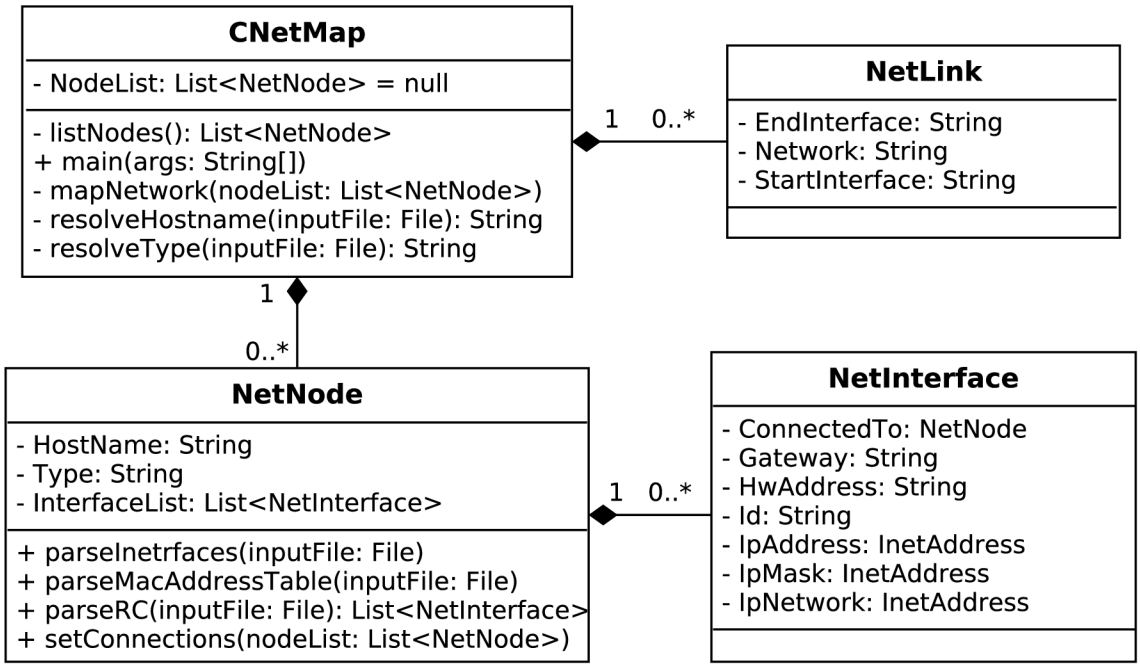
Jednotlivé funkcie programu sú teda klasifikované do tried a prvky reality sú modelované formou objektov. Každý objekt si bude pamätať svoj stav, ktorý je popísaný pomocou atribútov a s ostatnými objektmi bude komunikovať pomocou vlastných metód.

Návrh dátového modelu na konceptuálnej úrovni je vyobrazený na obr. 2.1 na strane 31.

Na najvyššej úrovni bude trieda `CNetMap`, ktoré bude zahŕňať metódy a dátové typy pre samotnú grafickú mapovaciu aplikáciu. Najdôležitejším atribútom tejto triedy bude zoznam sieťových uzlov typu `NetNode` a jeho počiatočná hodnota bude `null` (po spustení aplikácie je zoznam prázdny). Okrem hlavnej metódy na spustenie programu a metód obsluhujúcich grafické rozhranie aplikácie bude táto trieda obsahovať hlavne dve metódy určené na vyhľadanie jednotlivých uzlov skúmanej siete. Sú nimi metódy:

- `resolveHostname` – jediným vstupným parametrom bude textový súbor typu `running-config` a návratovou hodnotou bude reťazec obsahujúci hostiteľský názov skúmaného zariadenia, prípadne prázdny reťazec pri chybnom vstupnom súbore
- `resolveType` – vstupným parametrom je opäť textový súbor obsahujúci `running-config` a návratovou hodnotou reťazec obsahujúci typ zariadenia, môže nadobúdať hodnoty „switch“ alebo „router“.

Ďalej bude trieda obsahovať metódu `listNodes`, ktorej návratovou hodnotou bude zoznam nájdených sieťových uzlov, resp. hodnotu `null`, v prípade, že nebol nájdený žiaden uzol. Poslednou metódou bude `mapNetwork`, ktorá vykreslí samotnú mapu siete a jej vstupom je práve tento zoznam sieťových uzlov.



Obr. 2.1: UML diagram navrhnutej dátovej štruktúry

Po identifikovaní jednotlivých uzlov je už možné vytvoriť ich jednotlivé inštancie vlastného dátového typu – triedy `NetNode`. Medzi triedami `CNetMap` a `NetNode` bude použitý vzťah kompozície. Zrušením inštancie nadriadenej triedy (kontajneru) teda zaniknú aj všetky vytvorené inštancie triedy podradenej (elementu). Práve jedna inštancia triedy `CNetMap` bude obsahovať nula alebo viac inštancií triedy `NetNode`.

Trieda `NetNode` bude popisovať všetky potrebné vlastnosti na úrovni sieťových uzlov. Budú to nasledujúce atribúty:

- `HostName` – textový reťazec obsahujúci hostiteľský názov zariadenia
- `Type` – textový reťazec obsahujúci typ zariadenia, nadobúda hodnôt „switch“ alebo „router“
- `InterfaceList` – zoznam položiek typu `NetInterface`, predstavuje zoznam aktívnych rozhraní daného zariadenia

Trieda bude zahŕňať všetky metódy na spracovanie troch typov vstupných súborov (výpis `show running-config`, `show interfaces` a `show mac-address-table`). V nich budú metódy vyhľadávať využitím regulárnych výrazov všetky informácie potrebné k mapovaniu – IP adresy, fyzické adresy a brány jednotlivých rozhraní.

Ďalšou podriadenou triedou bude spomenutá trieda `NetInterface`, ktorá zahŕňa popis vlastností konkrétneho sieťového rozhrania. Medzi ňou a triedou nadradenou bude opäť vzťah kompozície, pričom práve jedna inštancia triedy `NetNode` bude obsahovať nula alebo viac inštancií triedy `NetInterface`. Táto trieda nebude obsahovať žiadne vlastné metódy. Atribútmi tejto triedy budú:

- `Id` – textový reťazec predstavujúci identifikátor sieťového rozhrania (napr. `Fa0/0`)
- `IpAddress` – IP adresa pridelená sieťovému rozhraniu
- `IpMask` – maska podsiete, do ktorej dané rozhranie patrí
- `IpNetwork` – adresa podsiete, do ktorej dané rozhranie patrí
- `HwAddress` – fyzická adresa tohto rozhrania
- `Gateway` – fyzická adresa brány, bude zaznamenávaná u uzlov typu prepínač, počiatočná hodnota nastavená na `null`
- `ConnectedTo` – inštancia triedy NetNode, predstavuje uzol, na ktorý je toto rozhranie pripojené

Spojenia medzi uzlami siete budú reprezentované jednoduchou triedou `NetLink`, ktorá bude uchovávať a poskytovať tieto atribúty:

- `StartInterface` – identifikátor počiatočného rozhrania tohto spojenia
- `EndInterface` – identifikátor koncového rozhrania tohto spojenia
- `Network` – adresa siete

Týmto sú už k dispozícii všetky potrebné informácie potrebné k zostaveniu mapy siete usporiadané hierarchicky a prehľadne v jednej dátovej štruktúre. Ďalším krokom bude navrhnúť postup, ktorým budú vstupné súbory spracované, všetky potrebné dáta zozbierané a následne bude nimi táto štruktúra naplnená.

2.3 Návrh algoritmu pre spracovanie vstupných súborov

Najdôležitejšou časťou navrhovaného programu bude úplné, správne a čo najefektívnejšie spracovanie vstupných súborov pre získanie všetkých informácií, ktoré naplnia dátovú štruktúru. Táto kapitola je venovaná návrhu a detailnému popisu algoritmov, ktoré budú túto funkciu vykonávať.

Program bude pre správne funkčné mapovanie topológie siete potrebovať ako vstup tri druhy konfiguračných súborov Cisco zariadení.

Zo súborov typu `running-config` budú získané hostiteľské názvy a typy zariadení, ďalej zoznam aktívnych sieťových rozhraní spolu s informáciami protokolu IP (adresa, maska siete a sieť).

Pre úspešné mapovanie prepínačov budú navyše potrebné výpisy príkazov `show interfaces` a `show mac-address-table`, ktoré obsahujú údaje z linkovej vrstvy modelu ISO/OSI, čiže fyzické adresy rozhraní a ich brán.

Spracovanie každého typu súboru je prehľadne znázornené na vývojových diagramoch (obrázky 2.2, 2.3 a 2.4) so stručným slovným popisom.

2.3.1 Súbory typu running-config

Z konfiguračných súborov bude najskôr dvomi procedúrami (`resolveHostname` a `resolveType`) naplnený zoznam uzlov v sieti, každému uzlu bude priradený zistený hostiteľský názov a typ zariadenia (viď obr. 2.2 na strane 34).

V cykle bude čítaný vždy nasledujúci riadok súboru, pokým ten nebude obsahovať definíciu hostiteľského názvu. Tento riadok bude následne rozdelený na jednotlivé slová a slovo za kľúčovým slovom `hostname` bude uložené do parametru `hostname` novovytvoreného objektu typu `NetNode`.

Po nájdení hostiteľského názvu bude druhá procedúra opäť čítať súbor po jednotlivých riadkoch. Ak bude aktuálny riadok obsahovať definíciu IP adresy a zároveň riadok predchádzajúci tomuto riadku nebude obsahovať definíciu virtuálneho rozhrania `Vlan`, vytvorí sa nová inštancia objektu `NetInterface` a uložia sa prečítané (IP adresa, maska) či spočítané (adresa siete) príslušné parametre. Počet aktívnych rozhraní tohto objektu sa zvýši o jeden.

Po dosiahnutí koncu súboru procedúra vyhodnotí typ zariadenia ako buďto prepínač – počet fyzických rozhraní s pridelenou IP adresou je rovný nule, alebo smerovač – počet fyzických rozhraní s pridelenou IP adresou je väčší než nula.

2.3.2 Súbory typu interfaces

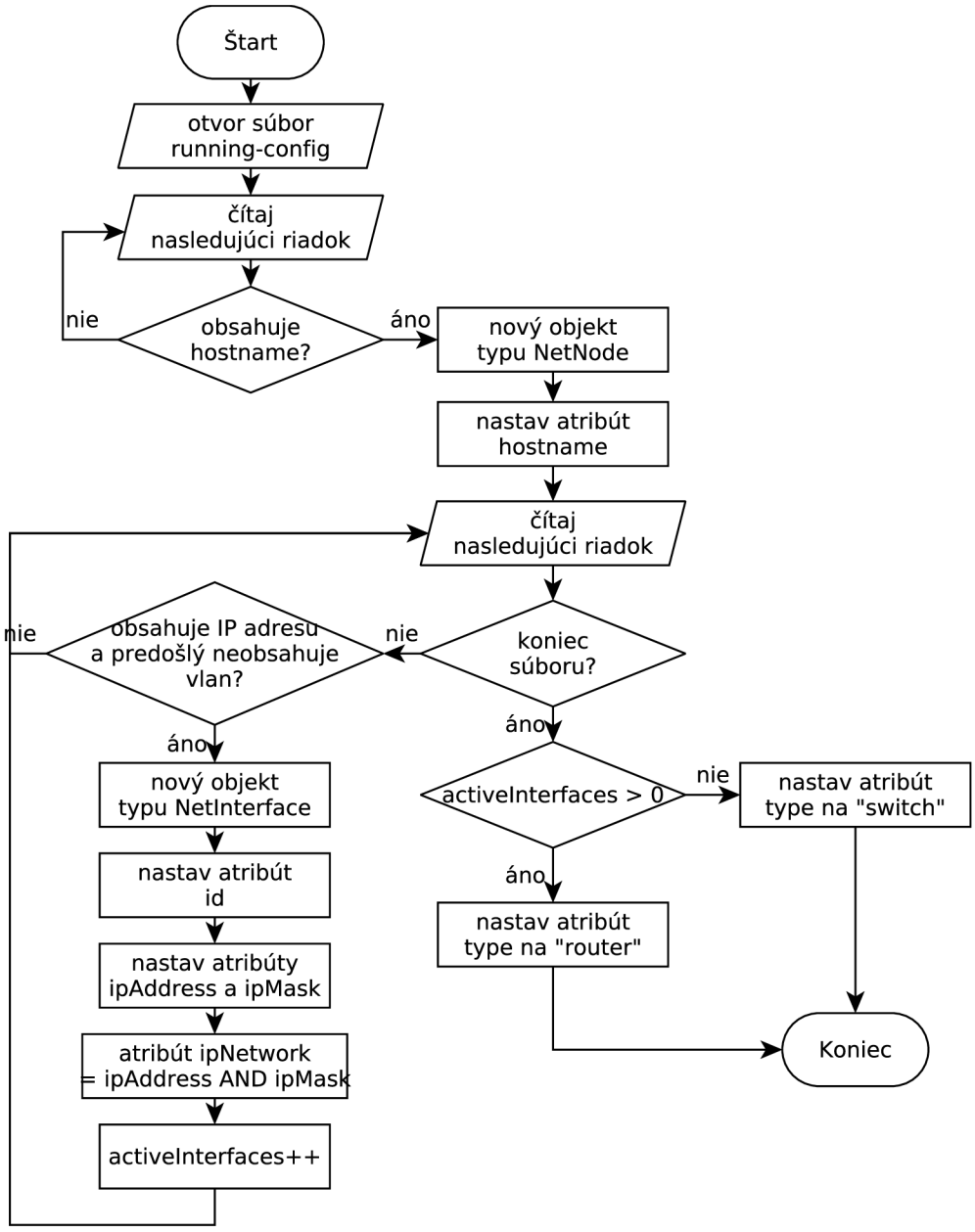
Vstupný súbor bude čítaný po riadkoch, až kým nebude nájdený riadok odpovedajúci definícii aktívneho rozhrania, napr.:

Výpis kódu 2.5: Časť výstupu príkazu show interfaces

```
GigabitEthernet0/0 is up, line protocol is up (connected)
Hardware is CN Gigabit Ethernet, address is 00d0.ba20.1201 (bia 00
   d0.ba20.1201)
Internet address is 10.0.40.1/30
```

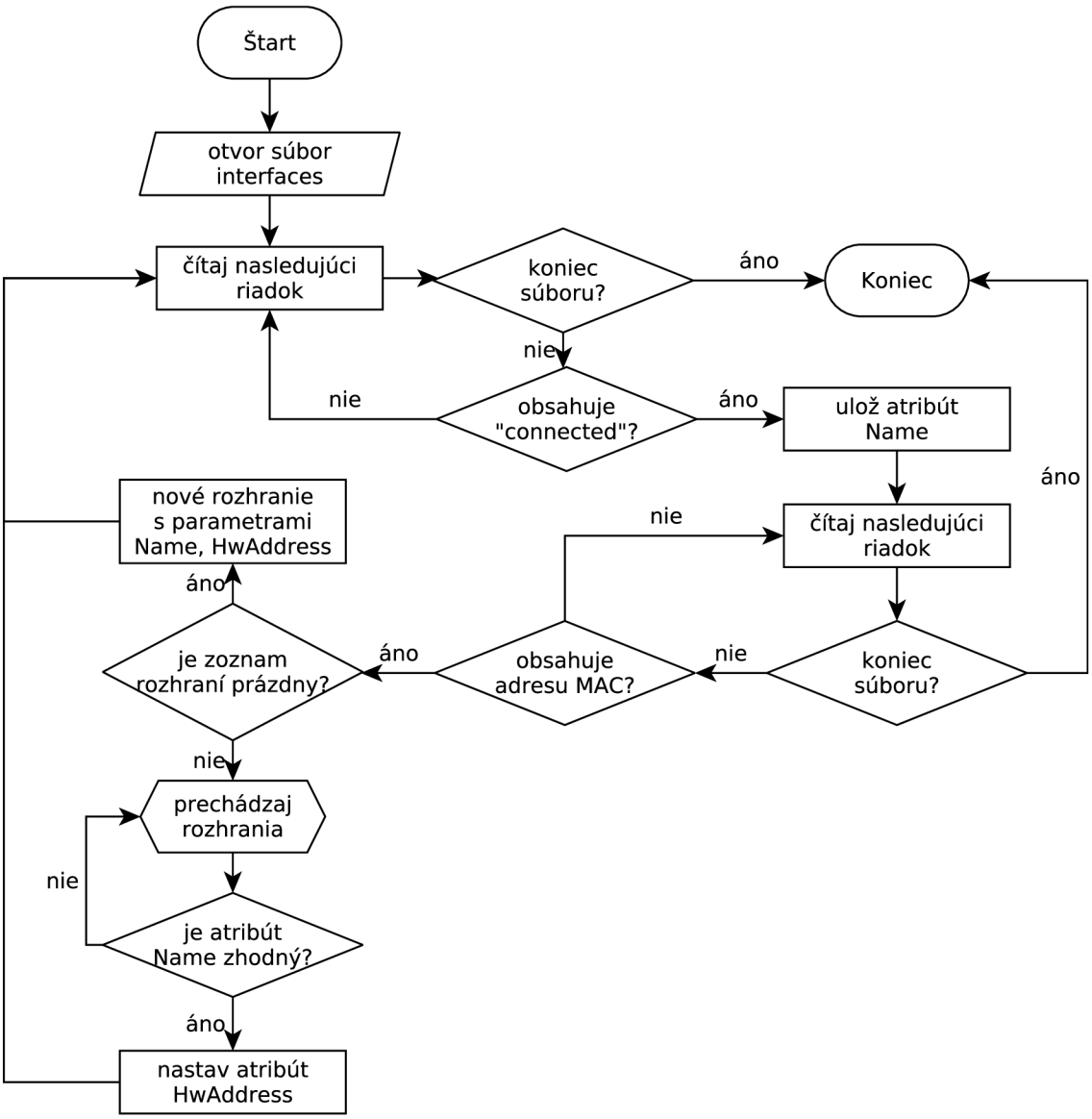
Z nasledujúceho riadku bude fyzická adresa uložená ako parameter `HwAddress` rozhrania triedy `NetInterface` s odpovedajúcim identifikátorom. Keďže rozhrania sú v rôznych typoch konfiguračných súborov označené rôznym spôsobom (celým názvom, napríklad `GigabitEthernet0/0` alebo skráteným identifikátorom, napríklad `Gi0/0`) je nutné celý názov rozhrania pred porovnávaním previesť na skrátený tvar. Na takýto prevod budú slúžiť metódy `toId` resp. `toName` z triedy `NetNode`. Metódy sú bližšie popísané v sekcii 3.2.

Algoritmus (znázornený na obr. 2.3 na str. 35) následne podľa toho, či už existuje záznam o rozhraním s daným identifikátorom buďto vyberie toto rozhranie a nastaví mu parameter `HwAddress` na hodnotu prečítanú zo súboru, alebo v prípade, že neexistuje, vytvorí nový objekt rozhrania s týmto identifikátorom a fyzickou adresou.



Obr. 2.2: Vývojový diagram spracovania dát zo vstupného textového súboru typu `running-config`

Algoritmus je ukončený po dosiahnutí koncu vstupného súboru.



Obr. 2.3: Vývojový diagram spracovania dát zo vstupného textového súboru typu `interfaces`

2.3.3 Súbory typu mac-address-table

Vstupný textový súbor bude opäť čítaný po jednotlivých riadkoch. V prípade, že dôjde k zhode s riadkom výpisu príkazu `show mac-address-table` (viď výpis kódu na strane 29), čo odpovedá riadku obsahujúcemu adresu MAC v tvare štvoríc znakov oddelených bodkami a regulárnemu výrazu

```
([0-9A-Fa-f]){4}\.([0-9A-Fa-f]){4}\.([0-9A-Fa-f]){4}.
```

Z rovnakého riadku sú potom uložené informácie o identifikátore rozhrania a adrese východzej brány pre toto rozhranie. Následne je kontrolované, či už rozhranie s rovnakým identifikátorom nebolo spracované, pretože zo štruktúry súboru vyplýva, že len prvý záznam pre každé jedno z rozhraní odpovedá priamo pripojenému zariadeniu.

V ďalšom kroku potom algoritmus rozhraniu priradí atribút nájdenej adresy pripojenej brány, prípadne vytvorí nový objekt rozhrania, ak pre daný identifikátor nebol nájdený žiaden záznam.

Algoritmus je opäť ukončený po dosiahnutí koncu vstupného textového súboru. Celý vývojový diagram algoritmu je znázornený na obrázku 2.4 na str. 37.

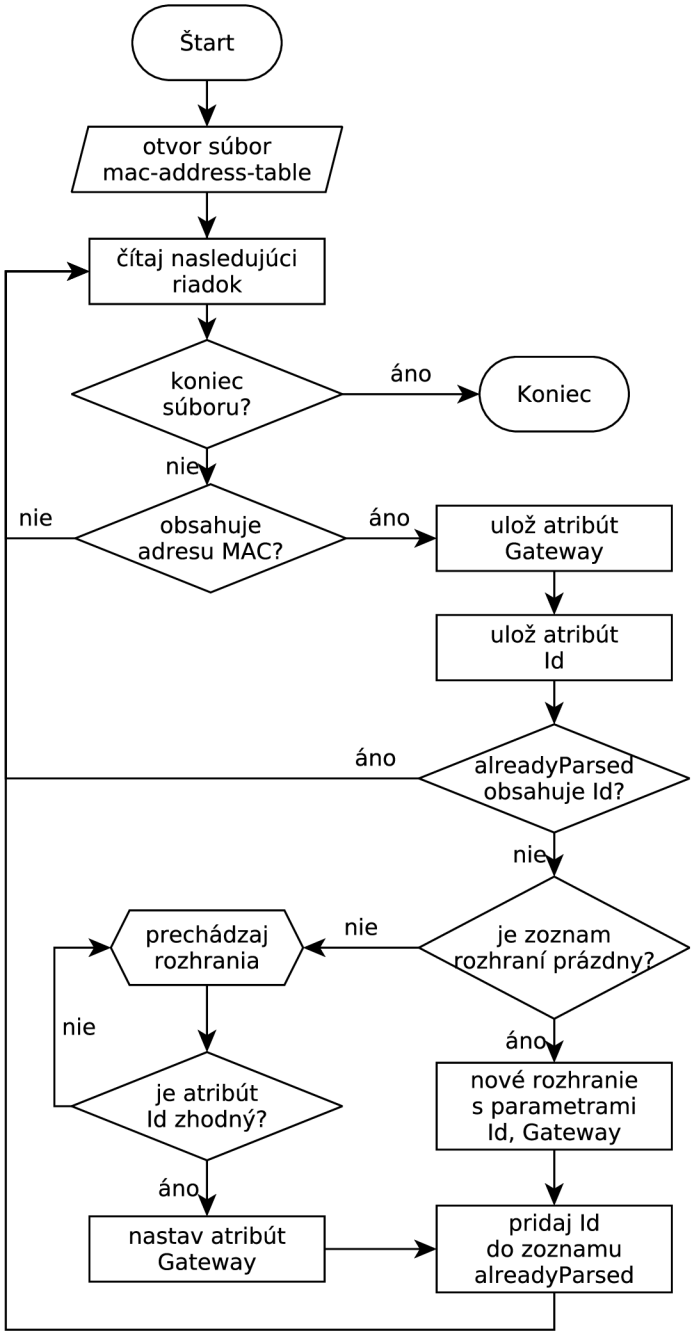
2.4 Návrh mapovacieho algoritmu

Tento algoritmus bude prebiehať spôsobom znázorneným vývojovým diagramom na obrázku 2.5 na strane 38. Cyklus bude postupne prechádzať zoznamom všetkých uzlov siete. U každého uzlu bude vyhodnotený typ zariadenia – prepínač alebo smerovač a algoritmus sa bude podľa tohto typu vetviť. Ak sa bude jednať o prepínač, pre každé jedno rozhranie z jeho zoznamu aktívnych rozhraní budú prehľadávané rozhrania všetkých ostatných sieťových uzlov. Pri každej takejto dvojici bude porovnávaná MAC adresa východzej brány lokálneho rozhrania s MAC adresou vzdialeného rozhrania.

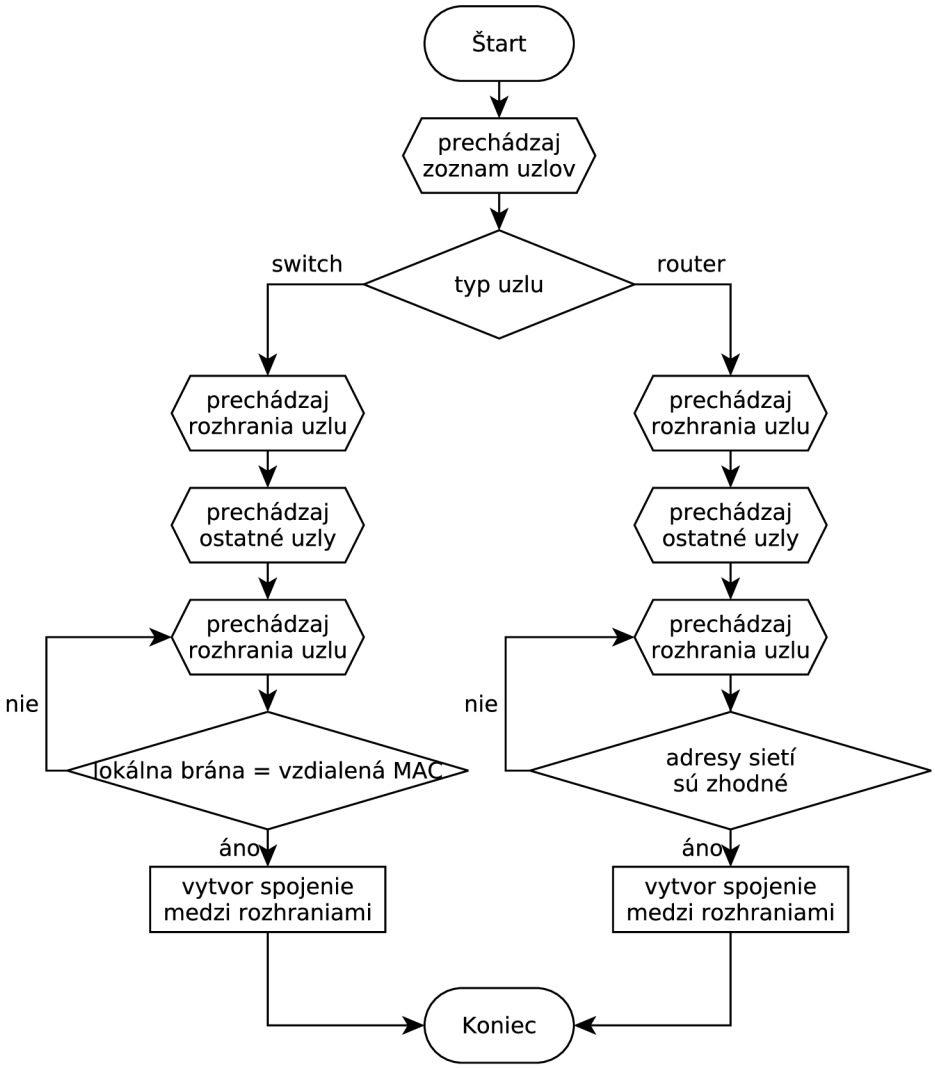
V prípade, že bude nájdená zhoda, medzi týmito dvomi rozhraniami bude vytvorené spojenie. To v praxi znamená, že oba objekty rozhraní si uložia ako atribút informáciu o protiľahlom pripojenom rozhraní. Cyklus takto pokračuje ďalej až pokým neprejde pre daný uzol celým zvyškom siete.

V situácii, kedy skúmaný uzol má typ smerovač, prebieha algoritmus obdobným spôsobom. Znovu sú prehľadávané rozhrania všetkých ostatných uzlov siete, ale namiesto fyzických adries budú porovnávané IP adresy siete lokálneho a vzdialeného rozhrania. Po nájdení zhody bude opäť medzi nimi vytvorené spojenie.

Po ukončení celého algoritmu bude mať každé aktívne rozhranie jednotlivých uzlov uloženú informáciu o protiľahlom pripojenom rozhraní. Keď sú tieto informácie známe, k vytvoreniu mapy bude už následne potrebné len reprezentovať každý uzol ako vrchol grafu, a medzi dvojicami, ktoré nesú informáciu o vzájomnom prepojení vytvoriť hrany grafu.



Obr. 2.4: Vývojový diagram spracovania dát zo vstupného textového súboru typu `mac-address-table`



Obr. 2.5: Vývojový diagram vytvárania spojení medzi jednotlivými uzlami siete

3 IMPLEMENTÁCIA VLASTNÉHO NÁVRHU

Aplikácia po zvolení vstupných textových súborov vytvorí a zobrazí zoznam nájdených Cisco zariadení, ktoré tvoria uzly skúmanej siete, identifikuje ich typ a aktívne sieťové rozhrania, nakoniec vytvorí spojenia medzi odpovedajúcimi rozhraniami a vykreslí samotnú mapu siete.

Pre implementáciu navrhnutej grafickej aplikácie bol zvolený programovací jazyk Java. Jedná sa o objektovo orientovaný programovací jazyk, čo zodpovedá a uľahčuje realizovanie konceptuálnemu návrhu aplikácie. Navyše je Java programovací jazyk multiplatformový, umožňuje po kompilácii spúšťať vytvorenú aplikáciu na ktorejkoľvek platforme, ktorá má k dispozícii interpret Javy – tzv. *Java Virtual Machine* (JVM).

Pri programovaní aplikácie bolo využitých niekoľko existujúcich tried jazyku Java, ich použitie riešenie problému výrazne zjednodušilo. Jedná sa o nasledujúce triedy:

- `javax.swing` – hlavná knižnica prvkov určených k ovládaniu programov pomocou grafického rozhrania. Obsahuje všetky základné elementy grafických rozhraní akými sú okná, tlačidlá, panely, ovládacie menu, tabuľky a podobne. Sú napísané kompletne v jazyku Java a sú tak nezávislé od platformy.
- `java.io` – knižnica poskytujúca prostriedky pre systémové vstupy a výstupy prostredníctvom dátových tokov. Umožňuje prístup programu k súborovému systému, čítanie a zápis do súborov.
- `java.util.List` – keďže program bude v bežnom prípade pracovať s väčším počtom inštancií jednotlivých objektov (či už sa jedná o uzly siete, sieťové rozhrania alebo spojenia) je vhodné združovať ich do zoznamov daného typu, práca so zoznamom je potom jednoduchšia ako s jednotlivými objektami samostatne.
- `java.util.regex` – táto knižnica zahŕňa triedy pre hľadanie zhody textu a vzoru definovaného regulárnym výrazom. V práci sú využívané k vyhľadávaniu konkrétnych postupností znakov vo vstupných textových súboroch.
- `edu.uci.ics.jung` – knižnica JUNG (*Java Universal Network/Graph Framework*) verzie 2, ktorá slúži na modelovanie, analýzu a vizualizáciu dát reprezentovatelných grafom alebo sieťou. V práci je využitá na vykreslenie mapy siete, automatické rozloženie uzlov v sieti a interakciu s grafom siete.

3.1 Metódy na spracovanie vstupných súborov

Tieto metódy spracúvajú vstupné súbory príslušného typu spôsobom popísaným vývojovými diagramami v sekcii 2.3 na strane 32. Pre čítanie textu u všetkých týchto metód bola použitá trieda `java.io.BufferedReader`, ktorá umožňuje efektívnejšie opakované čítanie súboru po riadkoch.

Pre prácu s regulárnymi výrazmi v programovacom jazyku Java je vhodné použitie dvoch tried z knižnice `java.util.regex` a to `Pattern` a `Matcher`. Pritom objekt typu `Pattern` bude predstavovať samotný regulárny výraz vo forme textového reťazcu a objekt typu `Matcher` textový reťazec, v ktorom bude zhoda hľadaná, v tomto prípade jeden riadok vstupného súboru.

Regulárne výrazy sú reťazce popisujúce celú množinu reťazcov. Môžu sa skladať z literálov, tj. explicitne zadaných znakov, ďalej tzv. metaznakov – špeciálnych symbolov zastupujúcich určitú množinu znakov (viď preddefinované skupiny znakov v tab. 3.2, str. 41) a kvantifikátorov (viď tab. 3.1, str. 41), určujúcich počet výskytov predchádzajúceho znaku alebo skupiny znakov. Okrem toho bude dôležitá ešte definícia hraníc:

- `^` – začiatok reťazcu (textu, v ktorom sa vyhľadáva)
- `$` – koniec reťazcu (textu, v ktorom sa vyhľadáva)

V metóde `resolveHostname` bude hľadaný reťazec odpovedať sekvencii: biele znaky, slovo „hostname“, biele znaky a ľubovolné znaky. To odpovedá regulárnemu výrazu

```
^\s*hostname\s*
```

V jazyku Java je ale znak spätného lomítka tzv. *escape* znakom, takže pre zapísanie znaku „\“ do vyhľadávacieho reťazcu ho je potrebné zdvojiť. Výsledný kód pre nájdenie príslušného riadku bude teda vyzerať nasledovne:

Výpis kódu 3.1: `CNetMap.java` nájdenie príslušného riadku textu

```
static final String PATTERN_HOSTNAME = "^\\s*hostname\\s*";
Pattern pattern = Pattern.compile(PATTERN_HOSTNAME);
do {
  line = buffer.readLine();
  match = pattern.matcher(line);
  if(match.find()) {
    String[] result = line.split("\\s");
    hostName = result[result.length-1];
    break;
  }
```

```
} while(line != null);
```

V metóde resolveType je postup rovnaký, použitými regulárnymi výrazmi však budú reťazce pre hľadanie riadkov s definíciou vlan rozhrania a jeho IP adresy:

```
^\s*interface\s*Vlan\s*
^\s*ip\s*address\s*
```

Tab. 3.1: Kvantifikátory v regulárnych výrazoch

Kvantifikátor	**Počet opakovaní**
?	minimálne 0 krát, maximálne 1 krát
*	minimálne 0 krát
+	minimálne 1 krát
{n}	práve n krát
{n,}	minimálne n krát
{m,n}	minimálne m krát, maximálne n krát

Tab. 3.2: Preddefinované skupiny znakov

`.`	ľubovolný znak
`\d`	číslice (`0-9`)
`\D`	ľubovolný znak okrem číslic
`\w`	znaky slova (`a-zA-Z0-9_`)
`\W`	ľubovolný znak okrem znakov slova
`\s`	biele znaky (medzera, tabulátor, zalomenie riadku)
`\S`	ľubovolný znak okrem bielych znakov

3.2 Pomocné metódy

Kód programu obsahuje aj pomocné metódy, ktoré doplňujú potrebnú funkcionalitu. V prípade mapovania s využitím údajov protokolu IP je najdôležitejším údajom adresa siete, pretože vo výsledku budú práve uzly s rovnakou adresou siete v mape navzájom prepojené. Táto adresa je získaná jednoduchým logickým súčinom adresy hosťa a masky podsiete. Tie sú v programe prečítané zo súboru a uložené ako typ `java.Inet4Address` (keďže sa jedná o IPv4 adresy). Nad týmto typom ale nie je možné prevádzať priame logické operácie, preto navrhnutá metóda `getIPSubnet` najskôr prevádza vstupnú adresu a masku typu `Inet4Address` na polia typu `byte`, čo je 8-bitový celočíselný typ v rozsahu -128 až 127.

Po tomto prevode je už možné vykonať logický súčin adresy a masky, výsledné pole typu `byte` je naspäť prevedené na typ `Inet4Address` metódou `getByAddress`, ktorej vstupný parameter je práve pole typu `byte`.

Výpis kódu 3.2: `NetNode.java` výpočet adresy siete

```
public static Inet4Address getIPSubnet(Inet4Address host,
    Inet4Address mask) throws UnknownHostException {
  Inet4Address subnet;
  byte[] byteHost = host.getAddress();
  byte[] byteMask = mask.getAddress();
  byte[] byteSubnet = new byte[4];

  for (int i = 0; i < 4; i++) {
    byteSubnet[i] = (byte) (byteHost[i] & byteMask[i]);
  }
  subnet = (Inet4Address) InetAddress.getByAddress(byteSubnet);
  return subnet;
}
```

Ďalšími pomocnými metódami sú metódy `toId(String name)` a `toName(String id)` určené pre vzájomný prevod medzi dvomi rôznymi formátmi značenia sieťových rozhraní v konfiguračných súboroch Cisco zariadení. Napríklad vo výpise `running-config` sú uvedené celé názvy rozhraní (napríklad `GigabitEthernet0/0`) a vo výpise `mac-address-table` skrátené tvary (napríklad `Gi0/0`). Aby bola identifikácia jednotlivých rozhraní jednoznačná, je potrebné vždy pracovať s jedným a tým istým tvarom značenia.

Pri prevode je využitá jednoduchá manipulácia s textovými reťazcami metódami `substring` – vráti reťazec, ktorý je podmnožinou vstupného reťazcu v určitom intervale a `concat` – metóda pre spájanie reťazcov. Prevod z celého názvu na skrátený identifikátor vyzerá nasledovne:

Výpis kódu 3.3: `NetNode.java` prevod názvov sieťových rozhraní na skrátené identifikátory

```
public static String toId(String name) {
  String prefix, suffix;
  prefix = name.substring(0, 2);
  if (prefix.equals("Se")) {
    suffix = name.substring(name.length() - 5);
  } else {
    suffix = name.substring(name.length() - 3);
  }
  return prefix.concat(suffix);
}
```

Pre opačný prevod je využitý rozhodovací prepínač (`switch`), ktorý rozlišuje tieto typy rozhraní: `Ethernet`, `FastEthernet`, `GigabitEthernet`, `Modem` a `Serial`. Zdrojový kód metódy je k nahliadnutiu na priloženom CD v zdrojovom súbore `NetNode.java`.

3.3 Vizualizácia mapy siete

Pri tvorbe programu bola na účely vizualizácie grafu – mapy siete použitá verzia 2 knižnice JUNG [7]. Základom tejto knižnice je objekt grafu `Graph<V, E>` definovaný v `edu.uci.ics.jung.graph`. Jeho rozhranie definuje niekoľko základných operácií, ktoré je možné nad grafom vykonávať. Jedná sa napríklad o:

- Pridávanie a odstraňovanie vertexov či hrán, získavanie celých kolekcií vertexov a hrán.
- Získavanie informácií o konečných bodoch grafu

Pre rôzne účely poskytuje knižnica nasledujúce typy grafov:

1. `DirectedGraph<V, E>` – graf s podporou len smerových hrán
2. `UndirectedGraph<V, E>` – graf s podporou len nesmerových hrán
3. `SimpleGraph<V, E>` – graf bez podpory paralelných hrán a vlastných smyčiek
4. `MultiGraph<V, E>` – graf s podporou paralelných hrán

V navrhnutom programe bude použitý nesmerový graf bez paralelných hrán (keďže sa jedná o model počítačovej siete). Inicializácia objektu grafu má nasledujúci tvar:

```
Graph<V, E> graph = new SparseGraph<V, E>(),
```

kde `V` je objekt reprezentujúci vertexy grafu a `E` objekt reprezentujúci hrany grafu. V tomto prípade budú vertexy typu `NetNode` a hrany typu `NetLink`. Konkrétny zápis pre vytvorenie inštancie grafu bude teda vyzerať takto:

```
Graph<NetNode, NetLink> graph = new SparseGraph<>()
```

Graf je po inicializácii prázdny, neobsahuje žiadne vertexy ani hrany. Vertex je možné do grafu pridať metódou `addVertex(V)`, pričom typ parameteru V musí súhlasiť s typom vertexov grafu, do ktorého je pridávaný.

Po pridaní potrebných vertexov do grafu je možné vytvoriť medzi nimi spojenia – hrany grafu. Na to je určená metóda `addEdge(E, V1, V2)`. Parameter `E` je pridávaná hrana grafu, parametre `V1` a `V2` sú počiatočným a koncovým vertexom tejto hrany. Typy opäť musia súhlasiť s typmi uvedenými pri inicializácii grafu.

Nižšie je uvedená kompletná ukážka vytvorenia a naplnenia grafu počítačovej siete. Po inicializácii grafu je v cykle pridaný každý uzol zo zoznamu uzlov `nodeList`. V ďalšom cykle sú postupne pridávané hrany grafu pre každú dvojicu prepojených sieťových rozhraní. Aby bolo zabránené vytváraniu duplicitných spojení, po prepojení je obom objektom rozhraní nastavený atribút `AlreadyConnected` na hodnotu `true`. V ďalších iteráciach cyklu budú teda už prepojené rozhrania preskočené.

Výpis kódu 3.4: `CNetMapUI.java` konštrukcia grafu a jeho naplnenie vertexmi a hranami

```
Graph<NetNode, NetLink> networkGraph = new SparseGraph<>();
for (NetNode nodes : nodeList) {
  networkGraph.addVertex(nodes);
}
for (NetNode nodes : nodeList) {
  for (NetInterface interfaces : nodes.getInterfaceList()) {
    if ((!interfaces.getAlreadyConnected()) && (interfaces.
   getConnectedNode() != null)) {
      NetLink link = new NetLink(interfaces.getId(), interfaces.
   getConnectedInterface().getId());
      networkGraph.addEdge(link, nodes, interfaces.getConnectedNode
   ());
      interfaces.setAlreadyConnected(true);
      interfaces.getConnectedInterface().setAlreadyConnected(true);
    }
  }
}
```

Správne vytvorenie a naplnenie grafu vertexmi a hranami je možné overiť metódou `toString()`, ktorej výstupom je textová reprezentácia daného grafu. Výstup môže vyzerať nasledovne:

Výpis kódu 3.5: Výstup metódy `toString()`

```
Vertices:router north,router east,router center
Edges:Se0/0/1  Se0/0/1[router center, router east] Se0/0/0  Se0
   /0/1[router east, router north] Se0/0/0  Se0/0/0[router
center, router north]
```

Ďalším krokom je grafická vizualizácia vytvoreného grafu. Na to budú potrebné dva nové objekty, a to:

- `Layout<V, E>` – rozloženie grafu, zabezpečuje mechanizmy pre automatické rozmiestnenie vertexov grafu podľa zvoleného algoritmu
- `VisualizationViewer<V, E>` – grafický komponent pre vykreslenie grafu, je ho možné umiestniť kdekoľvek v hrierarchii grafickej knižnice `Swing`.

Vo výsledku sú tak na vizualizáciu potrebné minimálne štyri komponenty:

1. samotný graf
2. rozloženie grafu `Layout<V, E>`
3. `VisualizationViewer<V, E>`
4. základný komponent grafického rozhrania, napríklad Swing rámec `JFrame`

Výpis kódu 3.6: `CNetMapUI.java` vizualizácia grafu

```
Layout<NetNode, NetLink> networkLayout = new ISOMLayout(
   networkGraph);
networkLayout.setSize(new Dimension(600, 440));
VisualizationViewer<NetNode, NetLink> vv = new VisualizationViewer
   <>(networkLayout);
vv.setPreferredSize(new Dimension(640, 480));
vv.setBackground(Color.white);

JFrame networkFrame = new JFrame("Network visualization");
networkFrame.setDefaultCloseOperation(JFrame.HIDE_ON_CLOSE);
networkFrame.getContentPane().add(vv);
networkFrame.pack();
networkFrame.setVisible(true);
```

V uvedenom príklade je použité rozloženie grafu ISOMLayout, ktoré implementuje algoritmus samoorganizačných grafov. Princíp takýchto algoritmov popísal B. Meyer [10].

Aby bol výsledný graf siete prehľadnejší, je možné jednotlivé vertexy vykresliť rozdielnou farbou na základe typu uzlu. K tomu je využitý objekt `org.apache.commons.collections15.Transformer`, ktorý v tomto prípade prevádza typ `NetNode` na typ farby `Paint`. Je nevyhnutné preťažiť metódu `transform`, aby správne rozhodovala podľa typu zariadenia vrátila požadovanú farbu.

Do vizualizácie grafu je zmena vo vykresľovaní farieb vertexov pridaná pomocou metódy `setVertexFillPaintTransformer`.

Výpis kódu 3.7: `CNetMapUI.java` kód pre vykreslovanie vertexov rôznymi farbami

```
Transformer<NetNode, Paint> vertexPaint = new Transformer<NetNode,
    Paint>() {
  @Override
  public Paint transform(NetNode node) {
    if (node.getType().equals(CNetMap.TYPE_SWITCH))
      return Color.GRAY;
    else if (node.getType().equals(CNetMap.TYPE_ROUTER))
      return Color.BLUE;
    else
      // chybný typ uzlu
      return Color.RED;
  }
};

vv.getRenderContext().setVertexFillPaintTransformer(vertexPaint);
```

Knižnica JUNG ďalej umožňuje užívateľovi interagovať s vyobrazeným grafom, a to najmä použitím myši a klávesnice. Pridaním nižšie uvedeného kódu bude vizualizácia grafu doplnená o nasledujúcu funkcionalitu:

- Po stlačení a podržaní ľavého tlačidla myši je možné graf posúvať.
- Po Stlačení klávesy `Shift`, ľavého tlačidla myši a jeho podržaní je možné grafom rotovať.
- Po Stlačení klávesy `Ctrl`, ľavého tlačidla myši a jeho podržaní je možné graf skosiť.
- Koliečkom myši je možné meniť priblíženie (mierku) grafu.

Tieto funkcie sú dostupné v tzv. transformačnom móde. Po prepnutí do módu výberu je naopak možné pomocou myši manuálne meniť pozíciu jednotlivých vertexov a meniť tak rozloženie grafu. Medzi týmito módmi je možné prepínať klávesami „T“ pre transformačný mód a „P“ pre mód výberu.

Výpis kódu 3.8: `CNetMapUI.java` pridanie podpory pre interakciu s grafom

```
DefaultModalGraphMouse gm = new DefaultModalGraphMouse();
gm.setMode(ModalGraphMouse.Mode.TRANSFORMING);
vv.setGraphMouse(gm);
vv.addKeyListener(gm.getModeKeyListener());
```

3.4 Grafické rozhranie

Súčasťou aplikácie je jednoduché grafické rozhranie (hlavné okno aplikácie je na obr. 4.3, str. 52). Bude sa skladať z niekoľkých základných elementov. Prvým z nich je hlavné menu aplikácie, využíva triedu `javax.swing.JMenu` a bude obsahovať položku pre ukončenie aplikácie triedy
`javax.swing.JMenuItem`.

Na hlavnom paneli aplikácie je rozmiestnených niekoľko tlačidiel určených k zadávaniu vstupných súborov. Aplikácia umožňuje zadávať vstupné súbory dvomi spôsobmi, automaticky a manuálne.

1. Automatický spôsob – po stlačení tlačidla *Import folder* môže užívateľ pomocou dialógu pre výber súborov zvoliť cestu k adresáru, v ktorom sa nachádzajú konfiguračné textové súbory zariadení tvoriacich skúmanú sieť. Tieto súbory však musia splňovať nasledujúce podmienky:
   - Súbory patriace k jednému zariadeniu musia mať spoločný názov a odlišovať sa len príponou súboru.
   - Prípona súboru určuje typ konfiguračného súboru. Pre typ `running-config` je určená prípona „.rc“, pre typ `interfaces` prípona „.if“ a pre typ `mac-address-table` prípona „.mat“.

   Súbory, ktoré nesplňujú tieto podmienky, nebudú správne alebo vôbec programom spracované.
2. Manuálny spôsob – užívateľ najskôr pomocou tlačidla *Add RC* pridá súbory typu `running-config`, nájdené zariadenia sa zobrazia v zozname pod týmto tlačidlom. Po označení uzlu v zozname je k nemu možné priradiť ďalšie súbory, a to buď typu `interfaces` tlačidlom *Add Interfaces* alebo typu `mac-address-table` tlačidlom *Add MAC*.

Ďalšie tlačidlá *Clear list* a *Map Network* sú určené k vymazaniu zoznamu uzlov, resp. k samotnému zobrazeniu mapy siete v novom okne.

Na vytvorenie dialógov pre výber vstupných súborov sú v programe použité objekty triedy `javax.swing.JFileChooser`. Umožňujú filtrovať výber súborov podľa ich prípony (viď obr. 4.2, str. 51).

Vyššie spomenutý zoznam uzlov využíva triedu `javax.swing.JList` a ako model dát triedu `NodeListModel`. Zoznam po naplnení obsahuje textové reťazce skladajúce sa z typu a hostiteľského názvu uzlu.

3.5 UML diagram výsledného programu

Na obrázku 3.1, str. 49 je znázornený UML diagram tried výsledného vytvoreného programu CNetMap. Pri porovnaní s návrhom na obr. 2.1 na strane 31 je vidno, že štruktúra programu prešla niekoľkými zmenami.

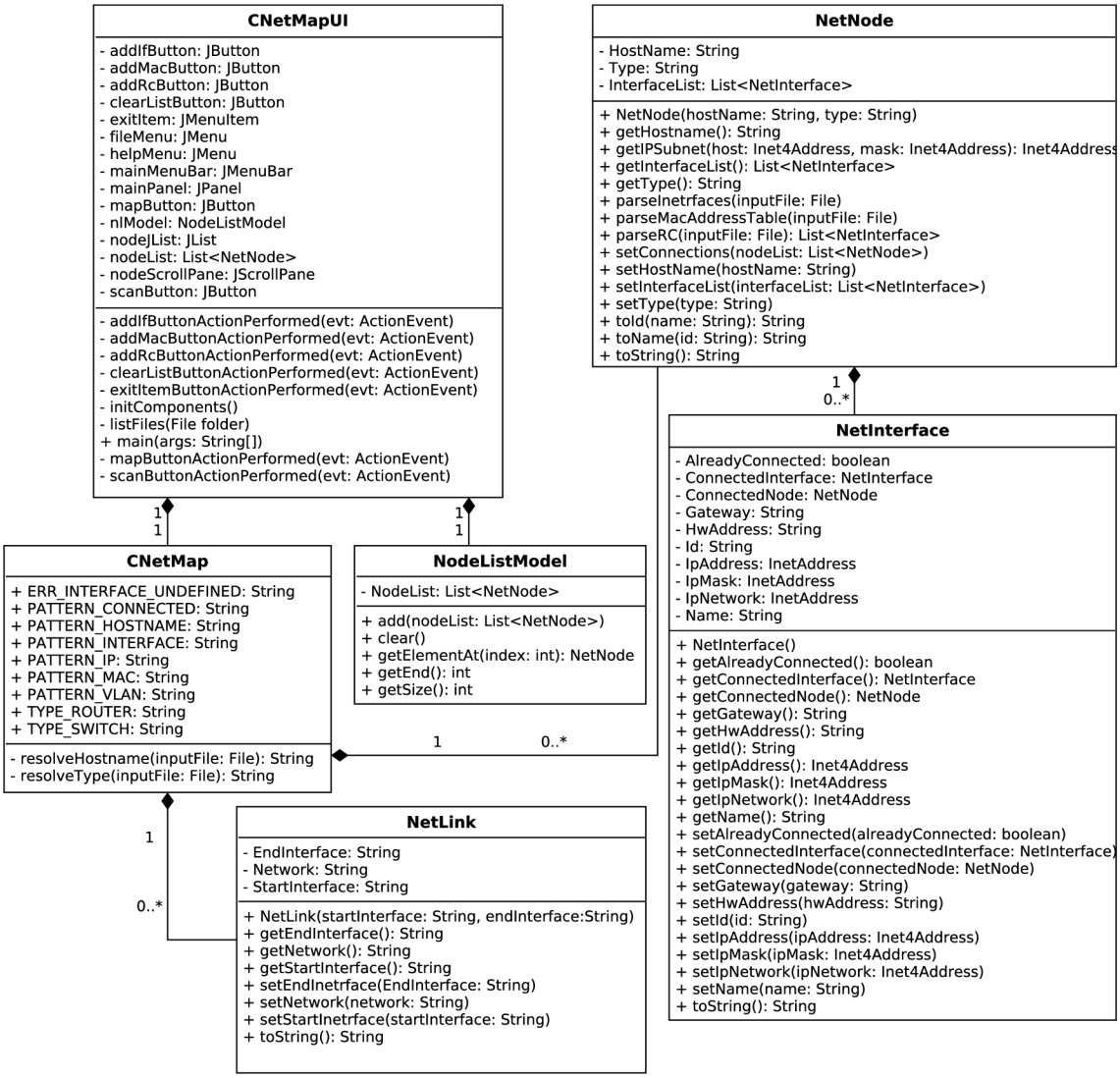
Triedou na najvyššej úrovni programu, obsahujúcou spustiteľnú metódu `main` je trieda `CNetMapUI`. Tá v sebe zahŕňa všetky objekty a metódy nejakým spôsobom obsluhujúce grafické užívateľské rozhranie aplikácie.

Ďalej je tu samotná metóda určená k vykresleniu grafu siete `mapButtonActionPerformed` spúšťaná po stlačení tlačidla `mapButton`.

Jej podradenou triedou je trieda `NodeListModel`, ktorá je rozšírením abstraktnej triedy `AbstractListModel` a zabezpečuje správny model dát pre grafický zoznam uzlov siete `nodeJList`.

Zvyšná časť dátového modelu v princípe súhlasí s teoretickým návrhom, ďalšou triedou je trieda `CNetMap` a podradené triedy `NetNode` (uchováva dáta o uzloch a metódy na spracovanie súborov), `NetInterface` (uchováva dáta o použitých sieťových rozhraniach) a `NetLink` (dáta liniek medzi uzlami).

Aj keď sa najdôležitejšie atribúty a metódy v návrhu a realizácii zhodujú, z porovnania UML diagramov je zrejmé, aké komplexné je oproti teoretickému návrhu konkrétne implementované riešenie problému.

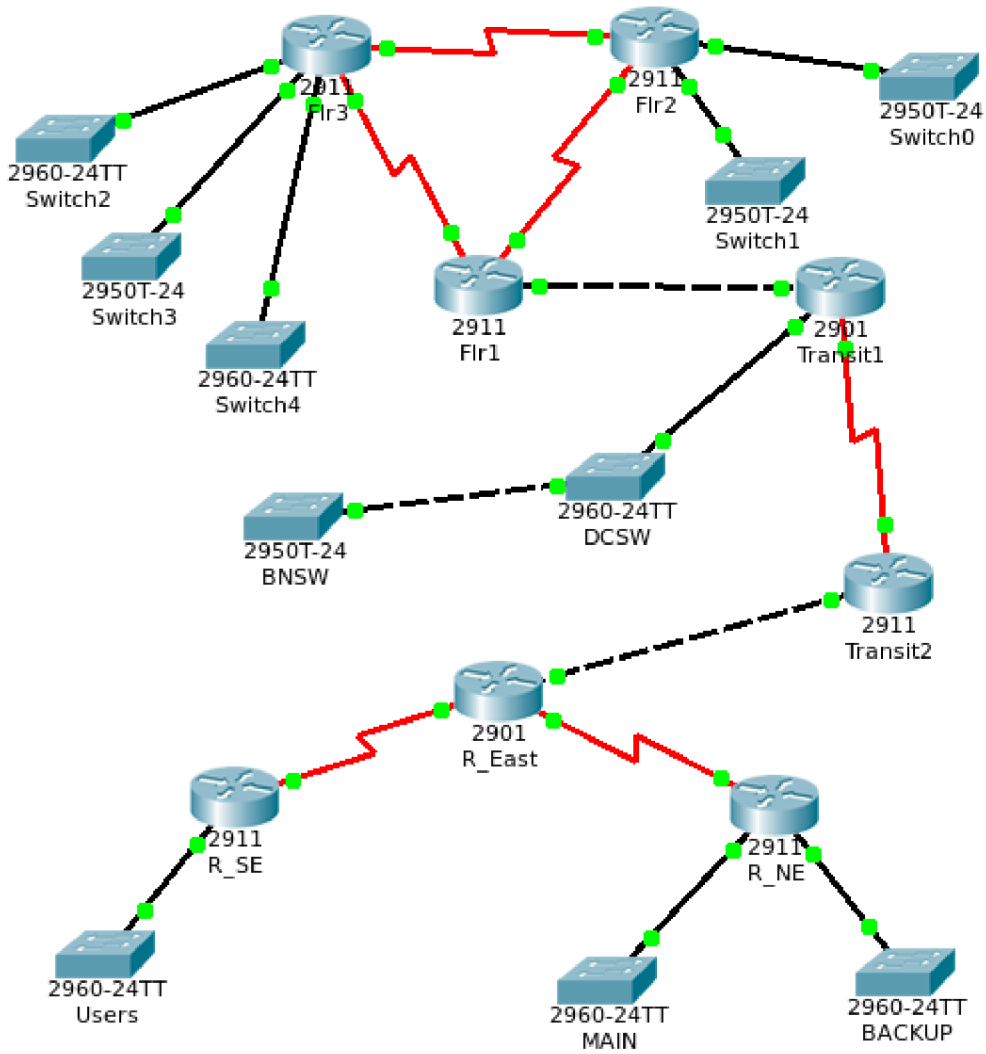


Obr. 3.1: UML diagram výsledného programu

4 VÝSTUPY NAVRHNUTÉHO PROGRAMU

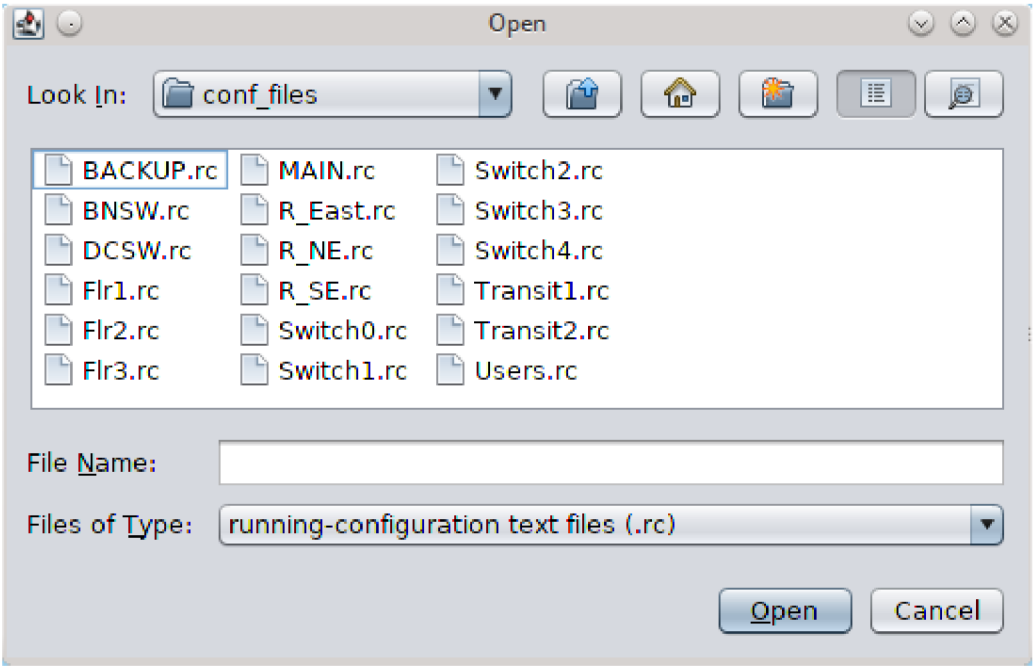
Po naprogramovaní a kompilácii navrhnutej aplikácie bola otestovaná jej funkčnosť. Ako testovacia sieť bola použitá malá virtuálna sieť vytvorená v programe *Cisco Packet Tracer*. Jedná sa o výkonný program určený k simulácii počítačových sietí založených na Cisco zariadeniach.

Topológia testovacej siete je vyobrazená na obr. 4.1, obsahuje osem nakonfigurovaných smerovačov (*Flr1*, *Flr2*, *Flr3*, *Transit1*, *Transit2*, *R_East*, *R_SE* a *R_NE*) a desať prepínačov (*Switch0* až *Switch4*, *BNSW*, *DCSW*, *Users*, *MAIN* a *BACKUP*).



Obr. 4.1: Topológia testovacej siete v programe Cisco Packet Tracer

Adresy všetkých aktívnych rozhraní u smerovačov boli manuálne staticky nakonfigurované, medzi smerovačmi bol nastavený smerovací protokol RIP. Konfigurácie všetkých týchto zariadení boli exportované do textových súborov pre spracovanie vytvorenou aplikáciou.



Obr. 4.2: Okno aplikácie pre výber vstupných textových súborov

Po spustení aplikácie (bola testovaná použitím Java Runtime Environment verzie 7) – z príkazového riadku príkazom

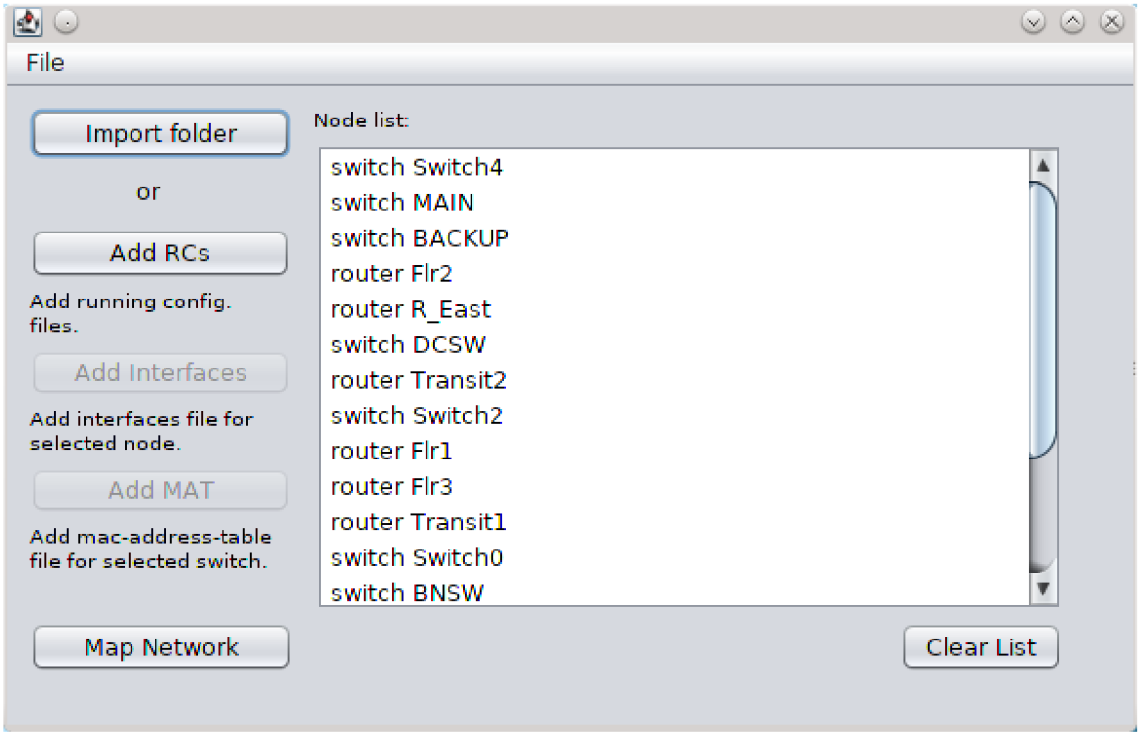
```
java -jar CNetMap.jar
```

v pracovnom adresári, kde sa binárny súbor nachádza, bola zložka so všetkými potrebnými vstupnými súbormi naimportovaná pomocou tlačidla *Import folder*. Okno pre výber súborov (viď obr. 4.2) umožňuje podľa potreby výber viacerých alebo len jedného súboru (adresára) zároveň a taktiež umožňuje filtrovať vstupné súbory podľa ich prípony (v prípade zadávania napr. súborov typu `running-config` prípona súboru *.rc).

Aplikácia následne správne zobrazí zoznam nájdených zariadení do zoznamu *Node list* v hlavnom okne (viď obr. 4.3, str. 52). Položky tohoto zoznamu tvorí dvojica typ zariadenia a hostiteľský názov zariadenia. Tlačidlom *Clear list* je možné zoznam uzlov v prípade potreby vymazať.

V prípade zadania chybných vstupných textových súborov nie sú do zoznamu zapísané žiadne identifikované zariadenia.

Po načítaní zoznamu sieťových uzlov je možné stlačením tlačidla *Map Network* zobraziť v novom okne samotnú mapu siete. Výsledná mapa je vyobrazená na obr. 4.4 na strane 53. Vertexy grafu boli pre lepšiu čitateľnosť rozmiestnené manuálne v móde výberu (dostupný po stlačení klávesy „P“).

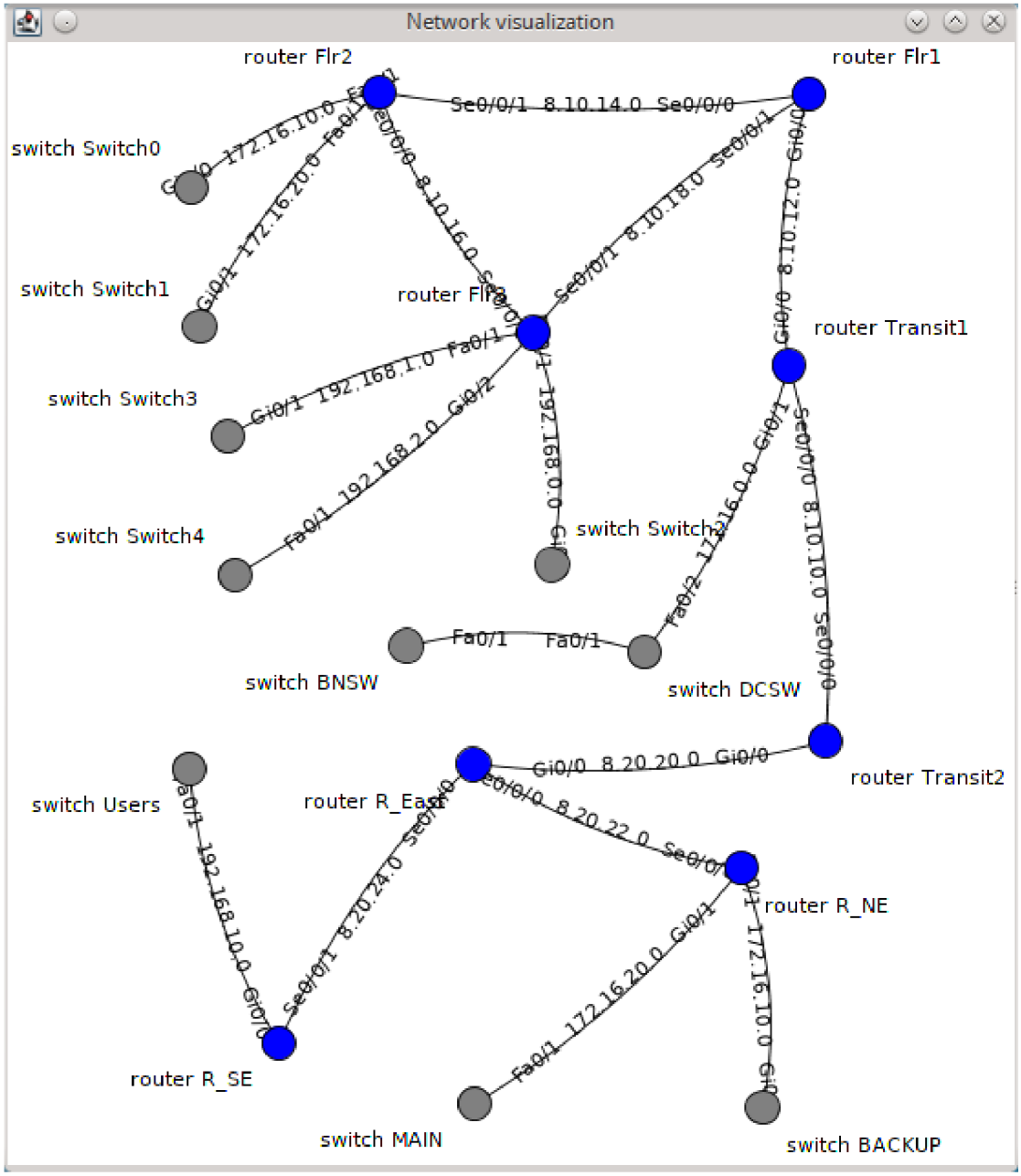


Obr. 4.3: Hlavné okno aplikácie

Je možné vidieť, že vertexy grafu predstavujúce sieťové uzly sú opäť označené popiskami tvorenými typom a hostiteľským názvom zariadenia. Typy zariadení sú navyše odlíšené rôznou farbou vertexu – modrou pre smerovače a sivou pre prepínače.

Hrany medzi vertexami predstavujúce sieťové spojenia sú označené popiskami obsahujúcimi identifikátory koncových sieťových rozhraní a adresu siete, do ktorej spojenie spadá.

Z porovnania obrázkov 4.1 a 4.4 jasne vyplýva, že topológie siete sa zhodujú, čiže navrhnutá aplikácia testovaciu sieť zmapovala správne.



Obr. 4.4: Topológia siete získaná programom CNetMap

ZÁVER

V rámci rešeršnej časti tejto práce boli preštudované a popísané princípy fungovania prepínačov a smerovačov a rôzne metódy použiteľné pri objavovaní a mapovaní sietí.

Praktická časť práce popisuje návrh a implementáciu programu pre mapovanie sietí. Pre získanie všetkých údajov potrebných k zostaveniu mapy siete nevyužíva aktívne sondovanie siete, ale boli využité len informácie z textových konfiguračných súborov použitých Cisco zariadení. Pri návrhu riešenia sa ukázalo ako nevyhnutné použitie troch typov vstupných súborov, sú nimi výpisy príkazov `show running-config`, `show interfaces` a `show mac-address-table` pre každé mapované zariadenie.

Z prvého súboru sú pritom získavané informácie o hostiteľských názvoch a typoch zariadení, zoznamoch ich aktívnych sieťových rozhraní a adresačných údajoch protokolu IPv4. Z ďalších dvoch súborov sú potom získavané informácie o fyzických adresách rozhraní, ktoré sú potrebné pre mapovanie prepínačov. Je to z dôvodu, že tie pracujú na linkovej vrstve modelu ISO/OSI.

Ďalším objaveným problémom pri návrhu bola situácia, kedy sa v skúmanej sieti vyskytuje viacero podsietí s rovnakou adresou siete z privátneho rozsahu adries. V takomto prípade nie je možné jednoznačne tieto podsiete rozlíšiť a výsledná mapa siete by neodpovedala skutočnosti. Výsledné riešenie teda počíta s prípadom, kedy minimálne podsiete medzi smerovačmi budú adresované vo verejnom rozsahu, a teda adresa každého rozhrania bude jedinečná. Podsiete medzi smerovačmi a prepínačmi môžu využívať aj privátny rozsah adries.

Bola navrhnutá a využitím programovacieho jazyku Java implementovaná aplikácia, ktorá zo spomenutých vstupných konfiguračných súborov načíta zoznam použitých Cisco zariadení tvoriacich sieť, určí a užívateľovi zobrazí ich hostiteľský názov a typ zariadenia.

Ďalej vykreslí mapu siete vo forme grafu, ktorého vrcholy predstavujú uzly siete, tie sú označené typom zariadenia a hostiteľským názvom, hrany grafu predstavujú spojenia medzi nimi a sú označené počiatočným a koncovým sieťovým rozhraním a adresou siete.

Všetky údaje potrebné pre mapovanie, akými sú hostiteľské názvy, IP adresy, fyzické adresy rozhraní, atď. boli vo vstupných súboroch vyhľadávané použitím regulárnych výrazov. Na vykreslenie mapy siete bola využitá knižnica JUNG určená na vizualizáciu grafov.

Program umožňuje zadávanie vstupných súborov buďto jednotlivo alebo hromadne, čo je výhodné najmä pri potrebe mapovania siete väčšieho rozsahu t.j. s väčším počtom uzlov.

Aplikácia bola otestovaná na virtuálnej sieti vytvorenej v prostredí programu

Cisco Packet Tracer, aplikácia následne zobrazila správny zoznam použitých zariadení s príslušnými hostiteľskými názvami a typmi. Takisto vizuálna mapa siete zodpovedala topológii navrhnutej testovacej siete.

Týmto je možné zhodnotiť, že vytvorený program pracuje podľa očakávania a všetky ciele práce, predložené v úvode práce, sa podarilo úspešne naplniť. Výsledné riešenie tejto práce je jedinečné, žiadne iné riešenia pracujúce na podobnom princípe neexistujú.

LITERATÚRA

[1] *802.1AB-REV – Station and Media Access Control Connectivity Discovery* [online]. posledná aktualizácia 30. 6. 2009, [cit. 10. 12. 2013] Dostupné z URL: <http://www.ieee802.org/1/pages/802.1AB-rev.html>.

[2] *802.1D – MAC bridges* [online]. posledná aktualizácia 26. 7. 2006, [cit. 10. 12. 2013] Dostupné z URL: <http://www.ieee802.org/1/pages/802.1D.html>.

[3] BANTSEEV, S. – LABBÉ I. *Traceroute Using an IP Option* [online]. Communications Research Centre, Ottawa, 2003, [cit. 14. 12. 2013] Dostupné z URL: <http://www.dtic.mil/cgi-bin/GetTRDoc?Location=U2&doc=GetTRDoc.pdf&AD=ADA465674>.

[4] *Cisco Discovery Protocol Configuration Guide* [online]. Cisco IOS Release 15M&T, [cit. 10. 12. 2013] Dostupné z URL: <http://www.cisco.com/en/US/docs/ios-xml/ios/cdp/configuration/15-mt/nm-cdp-discover.pdf>.

[5] HABRAKEN, J. – Hayden, M. *Sams Teach Yourself Networking in 24 Hours.* 3. vyd. Sams Publishing, 2004. 462 s. ISBN 0-672-32608-6 s. 41-45.

[6] *Introduction to Cisco IOS Software* [online]. Cisco Systems, Inc., Oct 31, 2003. [cit. 10. 5. 2014] Dostupné z URL: <http://jung.sourceforge.net>.

[7] *Java Universal Network/Graph Framework* [online]. [cit. 10. 5. 2014] Dostupné z URL: <http://jung.sourceforge.net>.

[8] LAMMLE, T. *CCNA Routing and Switching Study Guide 642-902 official certification guide.* John Wiley & Sons, Inc. Indianapolis, Indiana. 1178 s. ISBN 978-1-118-74961-6.

[9] *Load Balancing with CEF* [online]. [cit. 12. 5. 2014] Dostupné z URL: <http://www.cisco.com/en/US/products/hw/modules/ps2033/prod_technical_reference09186a00800afeb7.html>

[10] MEYER, B. *Self-Organizing Graphs and ISOM Layout* [online]. 8. 2. 1998, [cit. 10. 5. 2014] Dostupné z URL: <http://www.csse.monash.edu.au/~berndm/ISOM/>.

[11] ODOM, W. *CCNP Route 642-902 official certification guide.* Indianapolis: Cisco Press, 2010, xxxiv. 730 s. ISBN 978-1-58720-253-7. s. 140-143.

[12] *RFC 791 – Internet Protocol, DARPA Internet Program Protocol Specification* [online]. september 1981, [cit. 9. 12. 2013] Dostupné z URL: <http://tools.ietf.org/html/rfc791>.

[13] *RFC 1155 – Structure and Identification of Management Information for TCP/IP-based Internets* [online]. máj 1990, [cit. 9. 12. 2013] Dostupné z URL: <http://tools.ietf.org/html/rfc1155>.

[14] *RFC 1213 – Management Information Base for Network Management of TCP/IP-based internets: MIB-II* [online]. marec 1991, [cit. 9. 12. 2013] Dostupné z URL: <http://tools.ietf.org/html/rfc1213>.

[15] *RFC 1393 – Traceroute Using an IP Option* [online]. január 1993, [cit. 14. 12. 2013] Dostupné z URL: <http://tools.ietf.org/html/rfc1393>.

[16] *RFC 1631 – The IP Network Address Translator (NAT)* [online]. máj 1994, [cit. 10. 12. 2013] Dostupné z URL: <http://tools.ietf.org/html/rfc1631>.

[17] *RFC 1918 – Address Allocation for Private Internets* [online]. február 1996, [cit. 10. 5. 2014] Dostupné z URL: <http://tools.ietf.org/html/rfc1918>.

[18] *RFC 2328 – OSPF Version 2* [online]. apríl 1998, [cit. 9. 12. 2013] Dostupné z URL: <http://tools.ietf.org/html/rfc2328>.

[19] *RFC 4193 – Unique Local IPv6 Unicast Addresses* [online]. október 2005, [cit. 10. 5. 2014] Dostupné z URL: <http://tools.ietf.org/html/rfc4193>.

[20] *RFC 4251 – The Secure Shell (SSH) Protocol Architecture* [online]. január 2006, [cit. 12. 5. 2014] Dostupné z URL: <http://tools.ietf.org/html/rfc4251>.

[21] *RFC 4443 – Internet Control Message Protocol (ICMPv6) for the Internet Protocol Version 6 (IPv6) Specification* [online]. marec 2006, [cit. 10. 12. 2013] Dostupné z URL: <http://tools.ietf.org/html/rfc4443>.

[22] *RFC 5340 – OSPF for IPv6* [online]. júl 2008, [cit. 10. 12. 2013] Dostupné z URL: <http://tools.ietf.org/html/rfc5340>.

[23] *Secure Cisco Discovery Protocol* [online]. Cisco Discovery Protocol Configuration Guide, Cisco IOS XE Release 3S (Cisco ASR 1000), [cit. 8. 5. 2014] Dostupné z URL: <http://www.cisco.com/c/en/us/td/docs/ios-xml/ios/cdp/configuration/xe-3s/asr1000/cdp-xe-3s-asr1000-book/nm-cdp-secure.pdf>.

[24] SIAMWALLA, R. – SHARMA, R. – KESHAV, S. *Discovering Internet Topology*. Department of Computer Science, Cornell University, Ithaca, 1999. s. 2-8.

[25] SOSINSKY, B. *Networking Bible*. Indianapolis: Wiley Publishing, Inc., 2009. 890 s. ISBN 978-0-470-43131-3.

[26] TANENBAUM, A. S. *Computer Networks*. 4. vyd. Prentice Hall, New Jersey, 2002. 912 s. ISBN 9780130661029.

ZOZNAM SYMBOLOV, VELIČÍN A SKRATIEK

ARP Address Resolution Protocol

ATM Asynchronous Transfer Mode

CD Compact Disc

CDP Cisco Discovery Protocol

DHCP Dynamic Host Configuration Protocol

DoS Denial of Service

EEPROM Electrically Erasable Programmable Read-Only Memory

FDDI Fiber Distributed Data Interface

FIB Forwarding Information Base

ICMP Internet Control Message Protocol

IEEE Institute of Electrical and Electronics Engineers

IGMP Internet Group Membership Protocol

IOS Internetwork Operating System

IP Internet Protocol

ISO International Organization for Standardization

JUNG Java Universal Network/Graph Framework

JVM Java Virtual Machine

L2 Layer 2

LLDP Link Layer Discovery Protocol

LSA Link State Advertisement

LSU Link State Update

MAC Media Access Control

MD5 Message-Digest algorithm 5

MIB Management Information Base

NAT Network Address Translation

NMS Network Management System

NVRAM Non-volatile Random Access Memory

OSI Open Systems Interconnection

OSPF Open Shortest Path First

POST Power-on self test

RAM Random Access Memory

RFC Request for Comments

RIP Routing Information Protocol

ROM Read-Only Memory

SNMP Simple Network Management Protocol

SSH Secure Shell

TFTP Trivial File Transfer Protocol

TTL Time To Live

UML Unified Modeling Language

VLSM Variable-Length Subnet Mask

OBSAH PRILOŽENÉHO CD

1. Text práce v elektronickej forme
   `hlavny_dokument.pdf`
2. Zdrojové súbory programu CNetMap
   `CNetMap/src/CNetMap.java`
   `CNetMap/src/CNetMapUI.form`
   `CNetMap/src/CNetMapUI.java`
   `CNetMap/src/NetInterface.java`
   `CNetMap/src/NetLink.java`
   `CNetMap/src/NetNode.java`
   `CNetMap/src/NodeListModel.java`
3. Spustitelný binárny súbor programu CNetMap
   `CNetMap/dist/CNetMap.jar`
4. Adresár s externými knižnicami použitými v programe
   `CNetMap/dist/lib`
5. Adresár so vstupnými súbormi použitými pri testovaní programu
   `CNetMap/test`
6. Súbor projektu testovacej siete vytvorený v programe Cisco Packet Tracer
   `CNetMap/network_model.pkt`