Sem vložte zadání Vaší práce.

České vysoké učení technické v Praze
Fakulta informačních technologií
Katedra počítačových systémů

Bakalářská práce

Nástroj pro výuku adresace počítačových sítí

Jakub Kužel

Vedoucí práce: Ing. Viktor Černý

11. května 2015

Poděkování

Chtěl bych poděkovat Ing. Viktorovi Černému za vedení mé bakalářské práce a cenné rady. Děkuji také Veronice Kuželové a Barboře Kuželové za pomoc při gramatické kontrole práce.

Prohlášení

Prohlašuji, že jsem předloženou práci vypracoval(a) samostatně a že jsem uvedl(a) veškeré použité informační zdroje v souladu s Metodickým pokynem o etické přípravě vysokoškolských závěrečných prací.

Beru na vědomí, že se na moji práci vztahují práva a povinnosti vyplývající ze zákona č. 121/2000 Sb., autorského zákona, ve znění pozdějších předpisů. V souladu s ust. § 46 odst. 6 tohoto zákona tímto uděluji nevýhradní oprávnění (licenci) k užití této mojí práce, a to včetně všech počítačových programů, jež jsou její součástí či přílohou, a veškeré jejich dokumentace (dále souhrnně jen „Dílo“), a to všem osobám, které si přejí Dílo užít. Tyto osoby jsou oprávněny Dílo užít jakýmkoli způsobem, který nesnižuje hodnotu Díla, a za jakýmkoli účelem (včetně užití k výdělečným účelům). Toto oprávnění je časově, teritoriálně i množstevně neomezené. Každá osoba, která využije výše uvedenou licenci, se však zavazuje udělit ke každému dílu, které vznikne (byť jen zčásti) na základě Díla, úpravou Díla, spojením Díla s jiným dílem, zařazením Díla do díla souborného či zpracováním Díla (včetně překladu), licenci alespoň ve výše uvedeném rozsahu a zároveň zpřístupnit zdrojový kód takového díla alespoň srovnatelným způsobem a ve srovnatelném rozsahu, jako je zpřístupněn zdrojový kód Díla.

V Praze dne 11. května 2015 .....

České vysoké učení technické v Praze
Fakulta informačních technologií
© 2015 Jakub Kužel. Všechna práva vyhrazena.
Tato práce vznikla jako školní dílo na Českém vysokém učení technickém v Praze, Fakultě informačních technologií. Práce je chráněna právními předpisy a mezinárodními úmluvami o právu autorském a právech souvisejících s právem autorským. K jejímu užití, s výjimkou bezúplatných zákonných licencí, je nezbytný souhlas autora.

Odkaz na tuto práci

Kužel, Jakub. *Nástroj pro výuku adresace počítačových sítí.* Bakalářská práce. Praha: České vysoké učení technické v Praze, Fakulta informačních technologií, 2015.

Abstrakt

Tato bakalářská práce se zabývá vývojem multiplatformní aplikace, určené pro výuku adresace v počítačových sítích. Jejím cílem je také vybrat vhodný nástroj pro vytvoření grafického uživatelského rozhraní a dále nástroj pro vykreslení síťové topologie. Výsledkem této práce je funkční C++ aplikace, využívající knihovnu Qt pro grafické rozhraní a knihovnu QGV pro vykreslení síťové topologie.

Klíčová slova Qt, QGV, Graphviz, počítačová síť, Internet Protocol, IPv4, IPv6, adresace, směrování

Abstract

This bachelor thesis deals with development of an multiplatformal application intended for teaching on addressing in computer networks. It's also targeted on selection of an applicable tool for creation of a graphical user interface and then a tool for rendering network topology. The result of this thesis is a functional C++ application using the Qt library for graphical interface and QGV library for rendering network topology.

Keywords Qt, QGV, Graphviz, computer network, Internet Protocol, IPv4, IPv6, addressing, routing

Obsah

Úvod **1**

1 Analýza **3**
1.1 Cíl práce . . . 3
1.2 Požadavky na systém . . . 3
1.3 Adresace . . . 4
1.4 Směrování v IP sítích . . . 5
1.5 Existující řešení . . . 5
1.6 Použité technologie . . . 7

2 Realizace **17**
2.1 Uživatelské rozhraní . . . 17
2.2 Jádro . . . 22

3 Testování **25**
3.1 Testování OS . . . 25
3.2 Testování funkčnosti . . . 26
3.3 Instalační příručka . . . 28

Závěr **31**

Literatura **33**

A Seznam použitých zkratek **35**

B Obsah přiloženého CD **37**

Seznam obrázků

1.1 Ukázka QGV 14

2.1 Schéma GUI 19
2.2 Ukázka Qt Creatoru 20
2.3 Ukázka topologie 20
2.4 Ukázka směrovací tabulky 21
2.5 Ukázka tabulky adresace 22

Úvod

Počítačové sítě jsou dnes nejčastěji postavené na protokolu IP. Tento protokol je navržen zejména pro použití v propojeném systému více počítačových sítí. IP protokol zajišťuje přenos dat mezi jednotlivými zařízeními v celém systému, kde každé zařízení v tomto systému má přidělenou unikátní adresu sloužící k jeho jednoznačné identifikaci. Pro správné fungování protokolu je tedy nutné přidělit jednotlivým podsítím správné adresní prostory a vyplnit směrovací tabulky směrovačům propojujícím tyto sítě.

Tato problematika je probírána na začátku bakalářského předmětu Počítačové sítě (BI-PSI) na FIT ČVUT a k jejímu procvičení se zde mimo jiné využívá také online aplikace vytvořená na Fakultě elektrotechnické (`https://dsn.felk.cvut.cz/projects/psitest/`), ve které si může student vyzkoušet vyplnit adresní rozsahy a směrovací tabulky pro jednoduchou síť složenou z několika podsítí. Tato práce se zabývá vývojem podobného nástroje, který ale na rozdíl od předchozí aplikace nabídne širší možnosti generovaných testů a umožní také jejich parametrizaci.

V práci se kromě samotného popisu realizace věnuji také volbě vhodných nástrojů pro vytvoření grafického uživatelského rozhraní (GUI) aplikace a pro zobrazování síťové topologie. Výsledná aplikace je vytvořena pomocí programovacího jazyka C++ a využívá knihovnu Qt pro tvorbu GUI.

Jedním z požadavků na mou aplikaci je multiplatformnost, proto se v závěru této práce věnuji také testování aplikace na vybraných systémech a také přikládám instalační příručku, která popisuje postup instalace právě na tyto systémy.

Kapitola 1

Analýza

V této kapitole se nejprve zaměřím na popis zadání a cílů práce. Zároveň se v této části pokusím vysvětlit některé pojmy, které souvisí se zadaným tématem. Dále tuto práci porovnám s již existujícím řešením a na závěr popíši různé možnosti volby nástrojů a zdůvodním zvolené řešení.

1.1 Cíl práce

Cílem této bakalářské práce je vytvořit aplikaci pro výuku adresace a směrování v počítačových sítích. Výsledný nástroj umožní generovat a zobrazit graf síťové topologie na základě vstupních parametrů a pro tuto topologii vyplnit směrovací tabulku a adresy jednotlivých podsítí. Tyto údaje následně systém vyhodnotí podle zvolených kritérií.

Tato práce si dále klade za cíl vybrat vhodný nástroj pro vytvoření GUI a dále nástroj pro vykreslení vygenerované topologie.

1.2 Požadavky na systém

1.2.1 Funkční požadavky

- Generování síťové topologie na základě různých parametrů a její vykreslení.
  - Počet podsítí.
  - Počet routerů.
- Vkládání adres a masek pro všechny podsítě.
  - Adresa sítě
  - Maska
  - Nejnižší adresa stanice

  - Nejvyšší adresa stanice

- Kontrola očíslování podle různých kritérií.
  - Podle pořadí.
  - Podle velikosti.
  - Validní řešení.

- Práce s IPv4 i IPv6 adresami.

1.2.2 Nefunkční požadavky

- Použití programovacího jazyka C++.
- Multiplatformní aplikace.
- Jednoduché grafické rozhraní.

1.3 Adresace

Adresací se v počítačových sítích rozumí přidělení unikátních adres jednotlivým zařízením v rámci sítě. Tato adresa slouží jako jednoznačný identifikátor zařízení v síti, určený pro potřeby vzájemné komunikace všech zařízení.

1.3.1 Internet Protocol

Na síťové vrstvě se dnes nejčastěji používá protokol IP. Tento protokol je navržen zejména pro použití v navzájem propojených počítačových sítích. Protokol zajišťuje přenos bloků dat nazývaných datagramy od zdroje k cíli, kde zdroj a cíl jsou zařízení identifikovaná adresou fixní délky. [1]

Tato adresa je logicky rozdělena na dvě části, na část adresy sítě a část adresy zařízení. K tomuto rozdělení slouží maska sítě, která udává kolik bitů adresy (zleva) odpovídá adrese sítě. Zbylé bity odpovídají adrese zařízení a jednoznačně identifikují toto zařízení ve své síti. [2]

V současné době se používají dvě verze tohoto protokolu, IPv4 a IPv6.

1.3.1.1 IPv4

IPv4 je starší verze protokolu IP. IPv4 adresa má velikost 32 bitů a zapisuje se dekadicky po jednotlivých bytech, oddělených tečkou. Vzhledem k nedostatku volných IPv4 adres se dnes již postupně přechází na novější verzi protokolu IPv6.

1.3.1.2 IPv6

IPv6 je nová verze IP protokolu. Nejpodstatnější změnou oproti předchozí verzi je velikost IP adresy. IPv6 adresa má velikost 128 bitů a zapisuje se jako osm skupin čtyř hexadecimálních číslic oddělených dvojtečkou. [3]

1.4 Směrování v IP sítích

Jednotlivé stanice v rámci stejné sítě mohou společně komunikovat přímo, ale v případě komunikace stanic napříč různými sítěmi je třeba sítě propojit pomocí routerů.

Router propojuje dvě nebo více sítí a jeho rozhraní jsou tedy součástí těchto sítí a tak mají přidělené i příslušné IP adresy. Routery využívají adresu uvedenou v hlavičce datagramů k přenosu datagramů směrem k jejich cíli. Tento výběr cesty pro přenos se nazývá směrování.

Když router přeposílá datagram, tak musí určit, zda se cílová stanice nachází v síti přímo připojené k rozhraní routeru a datagram bude tedy zaslán přímo k cíli, nebo bude třeba zaslat datagram přes další router a v takovém případě určit přes který. Router nejprve porovná síťovou část adres všech rozhraní s odpovídající částí adresy cíle datagramu. Adresa sítě je popsána v sekci1.3.1. Pokud se u některého rozhraní nalezne shoda, tak to znamená, že se cílová stanice nachází ve stejné síti jako odpovídající rozhraní a datagram se tedy na toto rozhraní odešle.

Pokud se žádná shoda nenalezne, znamená to, že se cílová stanice nenachází v žádné přímo připojené síti a datagram bude tedy přeposlán do jiné sítě. Tato síť je vybrána na základě směrovací tabulky, která obsahuje seznam sítí a jim odpovídající rozhraní a adresy routerů na cestě k těmto sítím. To, zda datagram přísluší k některé síti, se porovnává stejným způsobem jako v předchozím případě. Směrovací tabulka obvykle obsahuje výchozí záznam, který se aplikuje vždy, když cílová adresa datagramu neodpovídá žádnému jinému záznamu ve směrovací tabulce. [4]

1.5 Existující řešení

Tato práce je inspirována již existujícím nástrojem, vytvořeným na Fakultě elektrotechnické, dostupným na adrese: `https://dsn.felk.cvut.cz/projects/psitest/`. Tento nástroj slouží k procvičování znalostí z adresace a směrování v počítačových sítích, ale je však příliš jednoduchý a neumožňuje žádnou parametrizaci generovaných topologií ani volbu kritérií pro hodnocení. V následujícím textu se budu zabývat všemi zásadními funkčními rozdíly mezi tímto nástrojem a mnou implementovaným řešením. Tím zároveň detailněji popíši očekávané vlastnosti a chování mé aplikace.

1.5.1 Generované topologie

První podstatný rozdíl se týká generovaných topologií. Spuštěním několika zkušebních testů lze vypozorovat, že původní systém generuje pouze omezený počet různých topologií v závislosti na počtu podsítí a routerů, jejichž počet nelze předem nastavit a který je náhodně čtyři nebo pět podsítí a tři nebo čtyři routery.

Generovaná topologie je vlastně strom, kde kořenový uzel představuje internet, do kterého je celá síť připojená. Z tohoto uzlu vede vždy jen jedna hrana do dalšího uzlu představujícího router na rozhraní internetu a připojených podsítí. Poté se již generuje asi šest různých struktur s tím, že každý uzel představující router má vždy jen jeden nebo dva potomky představující připojené sítě a jeho rodičovský uzel je vždy jiná podsíť, nikdy jiný router.

Mé řešení naproti tomu umožňuje nastavit počet generovaných podsítí v rozmezí tři až sedm. Počet routerů lze také nastavit, a to od jednoho routeru do tolika routerů kolik je nastaveno podsítí. To proto, že mé řešení stejně jako původní vždy generuje strukturu topologie jako strom, kde rodičem každého uzlu představujícího router je uzel představující síť a naopak a zároveň platí, že každý router má ve stromě pod sebou minimálně jednu síť. To tedy znamená, že není možné aby ve struktuře bylo více routerů než sítí. Pro tuto strukturu ale platí jen tato omezení, jinak je náhodná a proto je počet všech možných vygenerovaných struktur výrazně větší než u původního řešení.

1.5.2 Adresace

Test adresace je v původní aplikaci řešen pomocí tabulky, kde má uživatel za úkol vyplnit zvlášť pro každou podsíť adresu sítě, masku sítě, první a poslední adresu stanice a broadcast adresu. Údaj o počtu stanic každé podsítě, který uživatel pro správné vyplnění tabulky potřebuje, je generován náhodně pro každou síť a uživatel nemá možnost nastavit rozmezí.

Hodnocení probíhá pouze porovnáním uživatelského vstupu s jediným, předem vypočteným řešením. Uživatel musí podsítím přidělovat rozsahy adres v tom pořadí, v jakém jsou vypsány jednotlivé sítě v tabulce. Tyto rozsahy musí mít minimální možnou velikost podle počtu stanic a musí na sebe navazovat s minimálními možnými mezerami mezi jednotlivými rozsahy.

Mé řešení umožňuje uživateli nastavit interval počtu stanic pro generované sítě a umožňuje výběr mezi IPv4 a IPv6 typem adresy. Dále je možné si zvolit jedno ze tří kritérií pro hodnocení.

1.5.2.1 Kritéria hodnocení

1 První kritérium (podle pořadí) funguje v podstatě stejně jako u původního řešení, uživatel musí adresní rozsahy přidělovat sítím v tom pořadí, v jakém jsou uvedeny v tabulce, s minimálními mezerami a s minimální možnou velikostí.

2 Další kritérium (podle velikosti) vyžaduje, aby byly adresní rozsahy přidělovány v pořadí podle rostoucí velikosti sítí. Velikost jednotlivých rozsahů a mezer mezi nimi musí být opět minimální.

3 Poslední kritérium (validní řešení) vyžaduje jakékoliv správné řešení. V tomto případě se kontroluje pouze to, zda jsou jednotlivé sítě v požadovaném adresním rozsahu a zda se navzájem nepřekrývají.

1.5.3 Směrování

Test směrování spočívá ve vyplnění směrovací tabulky pro každý router z vygenerované topologie. V tabulkách se vyplňuje vždy pouze cílová síť a rozhraní následujícího routeru na cestě k této síti, takže se nevyplňují záznamy pro přímo připojené sítě. Směrovací tabulky se vyplňují tak, aby spolu mohly komunikovat všechny stanice a aby zároveň měly všechny stanice přístup k internetu. Tabulky musí být vyplněny v rostoucím pořadí čísla sítě a jako poslední musí být záznam pro výchozí cestu.

V této části se mé řešení příliš neliší, jediný rozdíl je, že záznamy v jednotlivých tabulkách mohou být vyplňovány v libovolném pořadí.

1.6 Použité technologie

Použití programovacího jazyka C++ je v tomto případě dáno požadavky na tuto práci. Je však třeba vybrat vhodné nástroje pro tvorbu GUI a pro vykreslování síťových topologií. V následujícím textu se zaměřím na popis nástrojů, které jsem při výběru zvažoval a rozeberu důvody volby konkrétních možností. Na závěr popíši práci se zvoleným nástrojem vykreslujícím síťovou topologii.

1.6.1 Grafické rozhraní

Při výběru nástroje pro tvorbu GUI jsem si měl podle zadání vybrat jednu ze tří knihoven: Tcl/Tk, GTK+ nebo Qt. V následujícím textu popíši tyto jednotlivé knihovny a zároveň odůvodním výběr zvoleného nástroje.

1.6.1.1 Tcl/Tk

Tcl je dynamický skriptovací programovací jazyk. Tcl je dostupný pro nejpoužívanější platformy jako je Unix, OS X, Android nebo Windows.

Tk je multiplatformní grafický toolkit vytvořený pro Tcl, ale může být použit i v mnoha jiných programovacích jazycích. [5]

V jazyce C++ lze využít knihovnu C++/Tk, která slouží jako rozhraní k Tk toolkitu. Nicméně tato knihovna pouze vytváří konkrétní příkazy, které se interpretují pomocí Tcl interpretru. To znamená, že C++/Tk knihovna je stále závislá na Tcl/Tk prostředí. [7]

1.6.1.2 GTK+

GTK+, nebo také GIMP Toolkit, je multiplatformní toolkit, určený pro vytváření GUI aplikací. GTK+ je napsaný v jazyce C, ale je dostupný i pro další programovací jazyky, například pro Python nebo Perl. GTK+ je svobodný software a je součástí GNU projektu. [6]

Pro GTK+ existuje řada nástrojů pro tvorbu uživatelského rozhraní, které umožňují navrhnout kompletní rozhraní pomocí grafického editoru bez nutnosti jeho programování. Nejpoužívanějším je nástroj Glade, který navržená rozhraní ukládá ve formě XML souboru, což umožňuje snadnou přenositelnost projektu mezi různými podporovanými programovacími jazyky. GTK+ dokáže vytvořit uživatelské rozhraní z těchto XML popisů. [7]

1.6.1.3 Qt

Qt je společně s knihovnou GTK+ jednou z nejpoužívanějších knihoven pro tvorbu GUI. Qt framework je komplexní multiplatformní framevork určený nejen k tvorbě aplikací s grafickým rozhraním, ale i k tvorbě konzolové aplikace či služby využívající pouze jeho jádro.

Pro nekomerční vývoj a pro studium je Qt nabízeno pod svobodnou licencí. V tomto případě je možné si zvolit mezi GNU GPL a GNU LGPL. Pro tvorbu komerčních aplikací nabízí Qt komerční licenci, která umožňuje vytvářet Qt aplikace, a to i s možností modifikace samotného Qt frameworku bez nutnosti zveřejnění zdrojového kódu.

Součástí Qt frameworku je i nástroj Qt Creator, který jako výše zmíněný Glade umožňuje navrhovat grafické rozhraní Qt aplikace a následně vytvořit odpovídající XML soubor. Z tohoto souboru je následně vygenerován zdrojový soubor, který lze zakomponovat do vlastního projektu.

Další užitečnou pomůckou z Qt frameworku je nástroj `qmake`. `Qmake` slouží ke generování makefile souborů. Makefile soubory jsou generovány na základě projektových souborů, které je možné vytvořit ručně, nebo je lze také vygenerovat programem qmake. Tento nástroj lze využít bez ohledu na to, zda se jedná či nejedná o Qt projekt. Stejně tak je možné vytvářet makefile soubory ke Qt projektům bez použití qmake. V takovém případě je ale nutné v makefile souboru před samotný překlad C++ kompilátorem vložit mezikrok, ve kterém se zdrojové soubory nejprve přeloží pomocí programu `moc`. `Qmake` toto provede automaticky, proto je pro Qt projekty vhodnější qmake využít.

Program `moc`, neboli Meta-Object Compiler, prohledává hlavičkové soubory projektu a v nich hledá třídy obsahující makro `Q_MAKE`. Z těchto souborů poté generuje nové C++ soubory s předponou `moc_`. To je nutné zejména proto, že Qt zavádí určitá rozšíření jazyka jako např. systém signálu a slotu. Běžný kompilátor jazyka C++ by tak původní zdrojový soubor nedokázal přeložit. [8] [9]

1.6.1.4 Výběr nástroje

S tvorbou GUI aplikací jsem neměl žádné předchozí zkušenosti a tak jsem podrobněji neznal ani žádný z uváděných nástrojů. Proto jsem se při výběru vhodného řešení řídil vlastnostmi jednotlivých nástrojů a jejich důsledky pro výslednou aplikaci. Vedlejším kritériem byl i osobní rozvoj, tedy volba nástroje, který bych se do budoucna rád naučil používat.

Hlavním požadavkem pro výběr byla podpora nejpoužívanějších operačních systémů a podpora nativního vzhledu výsledné aplikace na všech těchto platformách, neboli vzhledu nezávislého na GUI nástroji, ale závislého na konkrétním operačním systému a jeho nastaveních. Tento požadavek však splňují všechny tři nástroje.

Nástroj Tcl/Tk ale nenabízí takové možnosti jako Qt nebo GTK+. Práce s ním je sice jednoduší, ale právě na úkor jeho možností. Nástroje Qt a GTK+ jsou také oproti Tcl/Tk používanější a proto jsem se rozhodl, že si vyberu jeden z nich.

Oba nástroje Qt i GTK+ jsou vhodná řešení pro mou aplikaci a nepřišel jsem na žádnou vlastnost, která by vzhledem k požadavkům na grafické prostředí této aplikace jednoznačně zvýhodňovala jedno řešení oproti druhému. Proto jsem si s ohledem na vlastní preference vybral nástroj Qt, který mě zaujal také tím, že se nejedná pouze o nástroj k tvorbě GUI aplikací, ale nabízí například i podporu pro práci se sítí, databází, různými multimédii, nebo také třídu pro práci s vlákny, která umožňuje psát multiplatformní vláknové aplikace.

1.6.2 Vykreslování topologie

Další důležitou volbou byl výběr nástroje, který umožní přímo vykreslit síťovou topologii do okna aplikace. Jelikož si lze představit síťovou topologii jako graf, začal jsem pátrat po nástroji na vykreslování grafů, který by se dal snadno zakomponovat do aplikace a umožňoval vytvořit vlastní vzhled jednotlivých uzlů.

1.6.2.1 OGDF

Zajímavým řešením se zdálo být použití knihovny OGDF. OGDF je multiplatformní open-source C++ knihovna, zahrnující datové struktury a algoritmy, určené zejména pro použití v aplikacích vykreslujících grafy. Tato knihovna ale nezajišťuje přímo samotné vykreslení grafu a proto by se dala využít pouze pro určení pozic jednotlivých uzlů a hran a bylo by nutné zvolit další knihovnu pro samotné vykreslení struktury grafu. [10]

1.6.2.2 Igraph

Knihovna Igraph je další multiplatformní knihovna určená pro práci s grafy. Na rozdíl od knihovny OGDF nabízí Igraph kromě možnosti práce v programovacím jazyce C/C++ ještě verzi pro jazyky R a Python. Igraph stejně jako OGDF nabízí sadu algoritmů pro práci s grafy a také není určen přímo pro samotné zobrazení grafu. [11]

1.6.2.3 Graphviz

Graphviz je sada nástrojů určená pro vykreslování grafů. Narozdíl od obou předchozích knihoven nabízí Graphviz nástroje určené pouze pro rozvržení, případně samotné vykreslení grafu. Graphviz je možné použít dvojím způsobem.

První možností je využití přímo hotových programů. Tyto programy pracují s textovým vstupem popisujícím strukturu grafu v jazyce DOT. Jazyk DOT je speciální jazyk, určený pro popis grafů, který je jednoduchý a proto pohodlně čitelný pro člověka. Výstup může být potom v závislosti na parametrizaci například png obrázek vstupního grafu nebo prostý text popisující mimo jiné jednotlivé souřadnice uzlů a hran. Tento textový popis může být vhodný napřiklad pro další zpracování ve vlastní aplikaci. Jednotlivé programy jsou pojmenovány podle algoritmů, které používají k rozmístění uzlů a hran, např. program dot využívá stejnojmený algoritmus, který se snaží rozmisťovat jednotlivé uzly hierarchicky po vrstvách v jednom směru od kořenového uzlu. Tento způsob využití Graphvizu je možný i z vlastní C/C++ aplikace, a to například spuštěním těchto programů pomocí funkce system.

Druhou možností je využití Graphvizu jako C/C++ knihovny. Nástroj Graphviz umožňuje využít rozmisťovací algoritmy ve vlastní aplikaci podobně jako předchozí dva nástroje. K tomu slouží knihovna `cgraph`, která umožňuje sestavit graf přímo ve zdrojovém kódu voláním příslušných funkcí nebo pomocí vstupního souboru ve formátu DOT. [12]

1.6.2.4 Qanava

Qanava je C++ knihovna určená pro zobrazení grafu přímo v okně Qt aplikace. Knihovna Qanava kromě vykreslování grafů nabízí stejně jako předchozí nástroje různé možnosti rozmisťování uzlů. Tato knihovna je vyvíjena s Qt verze 5.4 a je primárně určena pro Qt Creator s použitím MSVC 2012, nicméně měla by fungovat i na Linuxu s použitím kompilátoru g++.

Vzhledem k tomu, že tato knihovna umožňuje jak automatické rozmístění uzlů tak i jejich vykreslení pomocí knihovny Qt a umožňuje vytvářet vlastní vzhled jednotlivých uzlů, se zdálo, že bude ideálním řešením pro tuto práci. Nicméně jsem se rozhodl, že prozkoumám ještě další možnosti, a to hlavně z důvodu, že na potřeby této práce je knihovna Qanava až příliš složitá a

nenabízí podle mého názoru dostatečně kvalitní dokumentaci. Jediná dokumentace, kterou jsem na stránkách projektu našel, je dokumentace zdrojového kódu. Kromě toho je na tomto webu už jen stručný popis základních vlastností a pár velmi jednoduchých příkladů použití. Předchozí nástroje naproti tomu nabízely kromě popisu jednotlivých tříd nebo struktur i manuál popisující práci s tímto nástrojem. Dalším důvodem pro hledání jiného nástroje bylo i to, že se mi nepodařilo spustit ani všechny testovací příklady, které byly součástí instalačního archivu. Při spouštění jednoho z nich docházelo stále k bezprostřednímu pádu aplikace. [13]

1.6.2.5 QGV

QGV je posledním nástrojem, který jsem testoval a pro který jsem se nakonec rozhodl. Tento nástroj stejně jako předchozí umožňuje jak rozmístění uzlů, tak samotné zobrazení v okně Qt aplikace. Pro výpočet pozic jednotlivých uzlů využívá QGV již zmíněnou knihovnu Graphviz a její algoritmus dot, který se pro účely této aplikace podle mého názoru zdá být nejvhodnější. [14]

QGV je velmi jednoduchá knihovna, která se skládá pouze z několika málo zdrojových a hlavičkových souborů. K tomuto nástroji sice neexistuje žádná dokumentace a součástí repozitáře je pouze jeden ukázkový příklad použití, ale vzhledem k jednoduchosti této knihovny nebyl problém z ukázky vypozorovat způsob jejího použití. Jednoduchost byla také důvodem proč jsem se pro toto řešení nakonec rozhodl.

Možnosti této knihovny téměř přesně odpovídají mým požadavkům, nicméně pro použití v mé aplikaci jsem musel provést určité úpravy. Nejdůležitější úprava spočívala v umístění popisku jednotlivých hran. Knihovna QGV nabízí pouze jedinou možnost popisku hran a to popisek se zarovnáním na střed hrany. Pro potřeby mé aplikace ale bylo nutné umožnit zarovnání spíše na oba konce hran, protože je třeba rozlišovat jednotlivá rozhraní routerů. To znamená, že v případě hrany směřující z routeru je potřeba zarovnání popisku na začátku hrany a naopak v případě hrany směřující do routeru. Vzhledem k tomu, že knihovna Graphviz, kterou QGV využívá, tuto funkcionalitu umožňuje a vzhledem k tomu, že samotné zdrojové kódy QGV jsou pro pochopení poměrně jednoduché, nebyla tato úprava velký problém.

1.6.2.6 Úprava QGV

Hlavní body této úpravy se nyní pokusím popsat v následujícím textu. Tento popis sice pro přehlednost doplňuji několika ukázkami ze zdrojového kódu, nicméně není jeho účelem vytvořit ucelenou představu o fungování QGV knihovny, ale spíše o rozsahu provedených změn. Kompletní upravená verze zdrojových kódů QGV knihovny je součástí přiloženého CD.

Nejprve bylo třeba upravit metodu `addEdge` ze třídy `QGVScene`, tato metoda je odpovědná za zavedení hrany mezi dvěma uzly a zároveň umožňuje

této hraně nastavit popisky. Hlavička upravené metody addEdge je vidět na následující ukázce.

```
QGVEdge * addEdge(QGVNode *, QGVNode *, const QString &, const
    QString &, const QString &);
```

Poslední tři argumenty této metody slouží k nastavení popisků na začátku, uprostřed a na konci hrany, původní verze této metody příjmala pouze jeden argument typu `QString`, který sloužil k nastavení popisku uprostřed hrany. Tato metoda původně volala pouze metodu `setLabel` ze třídy `QGVEdge`, jejímž prostřednictvím popisek hraně nastavila. Metodu `setLabel` jsem tedy přejmenoval na `setXLabel`, jelikož `xlabel` je atribut, který tato metoda nastavuje. Dále jsem přidal dvě nové metody `setHeadLabel` a `setTailLabel`.

```
void QGVEdge::setHeadLabel(const QString &label)
{
    setAttribute("headlabel", label);
}

void QGVEdge::setTailLabel(const QString &label)
{
    setAttribute("taillabel", label);
}
```

Tyto dvě nové metody se od původní metody `setLabel` liší pouze v atributu, který nastavují pomocí metody `setAttribute`. Jak metoda `setAttribute` funguje ilustruje následující ukázka.

```
void QGVEdge::setAttribute(const QString &name, const QString &value)
{
   agsafeset(_edge->edge(), name.toLocal8Bit().data(),
       value.toLocal8Bit().data(), "");
}
```

Funkce `agsafeset` je funkce Graphviz knihovny, jejíž deklarace se nachází v hlavičkovém souboru `cgraph`. Funkce slouží k nastavení atributů prvkům grafu. První argument této funkce je ukazatel na strukturu `Agedge_t`, který vrací metoda `edge`. Dalšími argumenty jsou pak textové řetězce reprezentující název atributu a hodnotu. Struktura `Agedge_t` obsahuje veškeré důležité informace pro sestavení hrany, a to včetně souřadnic pro umístění jednotlivých popisků a textové reprezentace těchto popisků. Využití této struktury znázorňuje následující úryvek kódu.

```
textlabel_t * head_label = ED_head_label(_edge->edge());
if(head_label)
{
   _head_label = head_label->text;
   _head_label_rect.setSize(QSize(head_label->dimen.x,
       head_label->dimen.y));
```

```
   _head_label_rect.moveCenter(QGVCore::toPoint(head_label->pos,
      QGVCore::graphHeight(_scene->_gragraph() )));
}

textlabel_t * tail_label = ED_tail_label(_edge->edge());
if(tail_label)
{
   _tail_label = tail_label->text;
   _tail_label_rect.setSize(QSize(tail_label->dimen.x,
      tail_label->dimen.y));
   _tail_label_rect.moveCenter(QGVCore::toPoint(tail_label->pos,
      QGVCore::graphHeight(_scene->_graph->graph() )));
}
```

`ED_head_label` a `ED_tail_label` jsou pouze makra, která vyberou ze struktury `Agedge_t` údaje o head a tail popisku, neboli o popisku začátku a konce hrany. Tyto údaje jsou reprezentovány strukturou `textlabel_t`. Ukázka výše pochází z metody `updateLayout` třídy `QGVEdge`. Tento úryvek kódu pouze načte text, souřadnice pro umístění a velikost obou přidaných popisků. Obdobný kus kódu obsahuje metoda `updateLayout` i pro původní středový popisek. Následující dva řádky přidané do metody paint tyto údaje využívají k vykreslení popisků do grafické scény.

```
painter->drawText(_head_label_rect, Qt::AlignCenter, _head_label);
painter->drawText(_tail_label_rect, Qt::AlignCenter, _tail_label);
```

1.6.2.7 Použití QGV

Knihovna QGV poskytuje pro práci s grafy čtyři jednoduché třídy, třídu `QGVScene`, `QGVNode`, `QGVEdge` a `QGVSubGraph`.

Třída `QGVScene` je odvozená od Qt třídy `QGraphicsScene`, která poskytuje plochu pro správu 2D grafických prvků. Třída `QGVScene` se stará o vkládání a správu uzlů, hran a podgrafů a o jejich vykreslení. Její použití ilustruje následující ukázka zdrojového kódu, která znázorňuje vytvoření jednoduchého grafu se dvěma uzly a jednou hranou. Výstup tohoto kódu je znázorněn na obrázku 1.1. Následující ukázka neznázorňuje pouze použití knihovny QGV, ale ukazuje i určité minimum pro vytvoření jednoduchého okna zobrazujícího obsah scény `QGVScene`. Proto je součástí ukázky i vložení potřebných hlavičkových souborů.

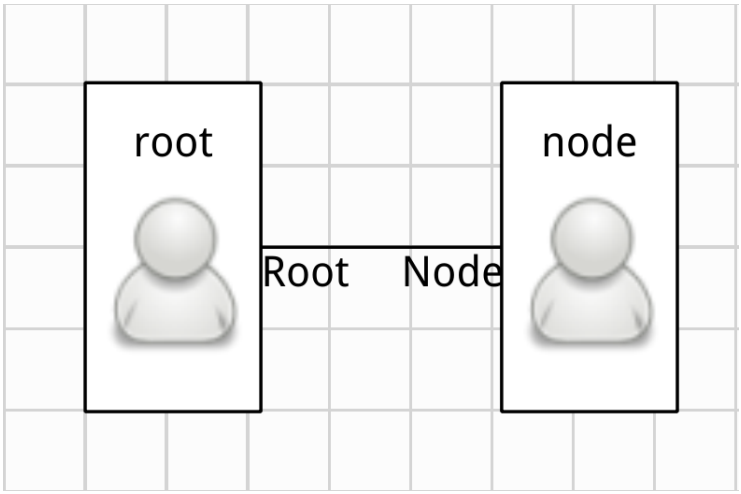
```
#include <QApplication>
#include <QGraphicsView>
#include "QGVScene.h"
#include "QGVNode.h"
#include "QGVEdge.h"

int main(int argc, char ** argv) {
```

```
    QApplication app(argc, argv);

    QGraphicsView graphView;
    QGVScene scene("graph");
    graphView.setScene(& scene);
    scene.setNodeAttribute("shape", "box");
    scene.setNodeAttribute("height", "1.4");
    scene.setNodeAttribute("style", "filled");
    scene.setNodeAttribute("fillcolor", "white");
    scene.setEdgeAttribute("minlen", "2");
    scene.setGraphAttribute("rankdir", "LR");
    QGVNode * root = scene.addNode("root");
    root -> setIcon(QImage("nodeIcon.png"));
    QGVNode * node = scene.addNode("node");
    node -> setIcon(QImage("nodeIcon.png"));
    scene.addEdge(root, node, "Node", "", "Root");
    scene.applyLayout();

    graphView.show();
    return app.exec();
}
```



Obrázek 1.1: Jednoduchý příklad použití knihovny QGV

Třída `QGraphicsView` z ukázky výše je třídou z knihovny Qt a poskytuje widget pro zobrazení obsahu `QGraphicsScene`, nebo z ní odvozených tříd, jako je třída `QGVScene`. Pro nastavení zobrazované scény slouží metoda `setScene`.

Následující metody `setNodeAttribute`, `setEdgeAttribute` a `setGraphAttribute` slouží pro nastavení atributů prvků v grafu, obdobně jako v nástroji Graphviz. Prvním argumentem metody je název atributu, který odpovídá názvům atributů z Graphvizu a druhým je příslušná hodnota. Narozdíl

od Graphvizu ale QGV neumožňuje tyto vlastnosti nastavovat jednotlivým uzlům a hranám zvlášť, ale pouze společně všem prvkům scény.

Metoda `addNode` slouží pro zavedení nového uzlu do scény. Jediným argumentem této metody je řetězec odpovídající názvu uzlu, který bude použit pro popis uzlu v zobrazovaném grafu. Metoda vrací ukazatel na objekt `QGVNode`, odpovídající právě vytvořenému uzlu. Díky tomu je možné následně těmto uzlům nastavit ikonu pomocí metody setIcon, která se bude zobrazovat uvnitř uzlu. Dále tyto ukazatele slouží k sestavení hran mezi jednotlivými uzly pomocí metody `addEdge`.

První dva argumenty metody `addEdge` jsou ukazatele na uzly typu `QGVNode`, které mají být hranou spojeny. Následující tři argumenty slouží k nastavení popisků hrany. První z těchto argumentů nastavuje popisek zarovnaný směrem k uzlu na konci hrany, další je zarovnán na střed hrany a poslední na začátek hrany. Původní verze knihovny QGV umožňovala zarovnat popisek hrany pouze na střed.

Kapitola 2

Realizace

Jednotlivé třídy aplikace jsou rozděleny na dvě skupiny, na třídy GUI a třídy jádra. Třídy jádra jsou odpovědné za operace související s vyhodnocením uživatelského vstupu nebo výpočty správného řešení. Pro předání výsledků mají tyto třídy definované rozhraní, které využívá GUI vrstva. GUI vrstva je odpovědná za grafickou prezentaci aplikace a tvoří rozhraní mezi aplikací a uživatelem.

V následujícím textu se často vyskytuje pojem widget. Widget je vizuální grafický prvek uživatelského rozhraní a je to základní stavební prvek všech aplikací s grafickým rozhraním. Qt nabízí základní widgety, jako jsou např. tlačítka, vstupní textová pole nebo popisky. Dále je možné tyto widgety skládat a uspořádat do složitějších, složených widgetů [9]. Každá třída v GUI vrstvě vytváří nějaký složený widget nebo je odvozena od nějakého základního Qt widgetu. V následujícím textu je pro popis grafického rozhraní používán jak pojem třída, tak pojem widget. V tomto případě mají tyto pojmy stejný význam, protože reprezentují stejný grafický prvek.

2.1 Uživatelské rozhraní

Uživatelské rozhraní je vytvořeno pomocí Qt frameworku. Součástí Qt frameworku je i nástroj Qt Creator, který slouží k návrhu kompletního uživatelského rozhraní pomocí relativně jednoduchého editoru bez nutnosti psaní zdrojových kódů. Já jsem se rozhodl, že při návrhu rozhraní své aplikace tento nástroj nevyužiji, protože požadované rozhraní není natolik složité a já jsem si chtěl vyzkoušet vytvořit rozhraní čistě pomocí programování v Qt knihovně. Toto rozhodnutí ale nakonec vedlo k poměrně nepřehledným zdrojovým souborům vytvářených widgetů. Tato nepřehlednost vznikla hlavně v rozsáhlých konstruktorech, které musí obsahovat veškerá nastavení pro daný widget.

Proto jsem se rozhodl, že si vyzkouším i práci v Qt Creatoru a jeden widget, konkrétně widget zobrazující nastavení testu, jsem se v něm pokusil předělat. Konečný návrh widgetu je ukázán na obrázku 2.2. Díky tomu je zdrojový

kód této třídy mnohem přehlednější a zároveň v něm lze snadno přistupovat k prvkům vygenerovaného widgetu pomocí jediného ukazatele.

2.1.1 Struktura kódu

Uživatelské rozhraní je tvořeno hlavním oknem reprezentovaným třídou `MainWindow`. Tato třída je odvozená od Qt třídy `QWidget` a je odpovědná za správu a zobrazování ostatních widgetů. Třída obsahuje atribut `stack` typu `QStackedWidget *`, který umožňuje vložení dalších tříd/widgetů a poté, pomocí metody `setCurrentWidget`, umožňuje přepínat mezi zobrazením jednotlivých těchto widgetů. V tomto případě je tedy použit pro přepínání mezi widgetem s úvodním nastavením testu a widgetem reprezentujícím samotný test. Následující úryvek kódu je z deklarace třídy `MainWindow` a znázorňuje její atributy:

```
private:
   QStackedWidget * stack;
   SettingsWindow * settings;
   TestWindow * test;
   QScrollArea * testArea;
```

Widget `stack` je hlavním widgetem třídy `MainWindow` a spravuje další dva widgety `settings` a `testArea`. Widget `settings`, který tvoří úvodní obrazovku s nastavením testu, je popsán v následující sekci 2.1.2. Widget `testArea` je typu `QScrollArea *` a obsahuje widget test, který se stará o zobrazení samotného testu. `TestArea` pouze vytváří oblast, do které widget test vkládá a která v případě, že její obsah přesáhne velikost okna aplikace, zobrazí vertikální i horizontální posuvníky pro pohyb po celém obsahu plochy. To proto, že widget test má proměnlivou velikost v závislosti na nastavení testu a na menších monitorech by se v některých případech nezobrazil celý.

Widget `test` je další složený widget typu `TestWindow *` a jeho atributy znázorňuje následující úryvek z deklarace třídy `TestWindow`.

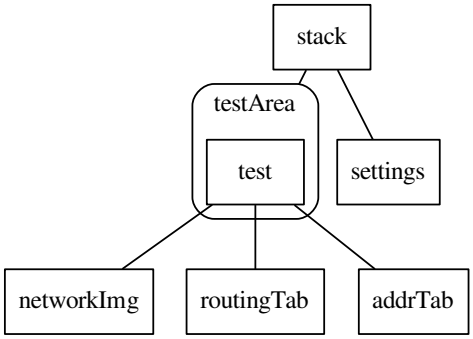
```
private:
   NetworkImage * networkImg;
   RoutingTable * routingTab;
   AddressTable * addrTab;
```

Třídy `NetworkImage`, `RoutingTable` a `AddressTable` tvoří obsah widgetu test a jsou popsány v následujících sekcích (2.1.3 - 2.1.5).

Struktura vazeb mezi jednotlivými prvky GUI, která je popsána v předchozím textu, je pro přehlednost znázorněna na obrázku 2.1.

2.1.2 Úvodní nastavení

Okno s nastavením je první okno, které se zobrazí po spuštění aplikace. Toto okno je tvořeno třídou `SettingsWindow`, jejíž grafický návrh jsem vytvořil



Obrázek 2.1: Znázornění vazeb mezi jednotlivými prvky GUI

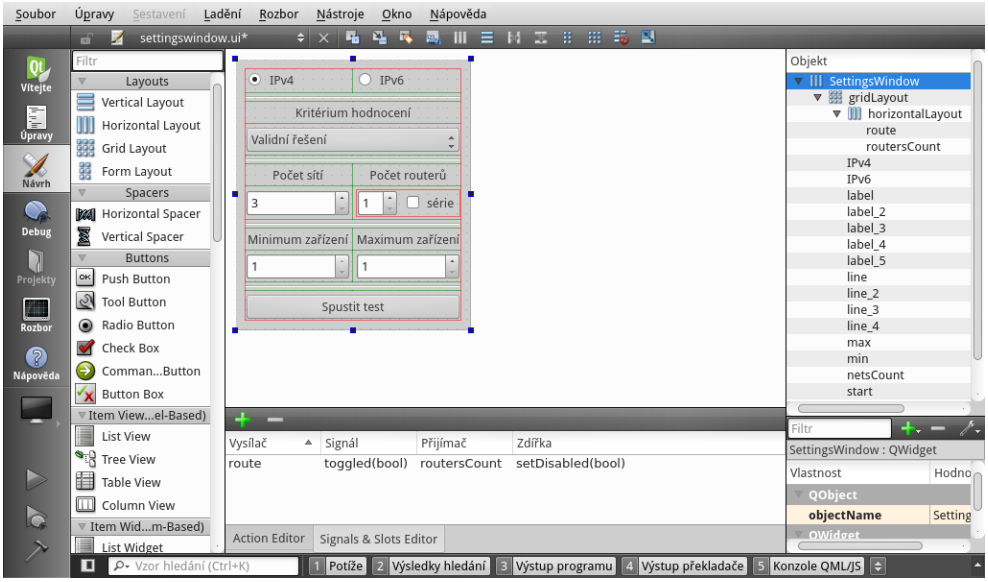
pomocí nástroje Qt Creator. Návrh této třídy v programu Qt Creator je znázorněn na obrázku 2.2.

Třída `SettingsWindow` zprostředkovává vstupní widgety pro nastavení typu IP, kritéria hodnocení, počtu sítí a routerů, minima a maxima počtu stanic a tlačítko pro spuštění testu. Widgety pro nastavení minima a maxima stanic jsou spojeny pomocí systému signálu a slotu tak, že změna jejich hodnoty nastaví druhému widgetu minimální nebo maximální hranici. To znamená, že nemůže dojít k situaci, kdy by hodnota minimálního počtu stanic byla větší než hodnota maximalního počtu, nebo naopak. Stejným způsobem funguje dvojice vstupních widgetů pro nastavení počtu sítí a routerů. Zde musí být, z důvodů popsaných v sekci 1.5.1, alespoň takový počet sítí, jaký je počet routerů.

2.1.3 Vykreslení topologie

Vykreslení síťové topologie má na starosti třída `NetworkImage`, která k tomu využívá upravenou knihovnu QGV. Pro vykreslení topologie pomocí knihovny QGV je třeba do widgetu vložit třídu `QGraphicsView`, která scénu vytvořenou touto knihovnou zobrazí do okna aplikace. V tomto případě není použita přímo třída `QGraphicsView`, ale z ní odvozená třída `NetGraphicsView`, která navíc umožňuje zvětšovat a zmenšovat scénu pomocí klávesy CTRL a kolečka myši.

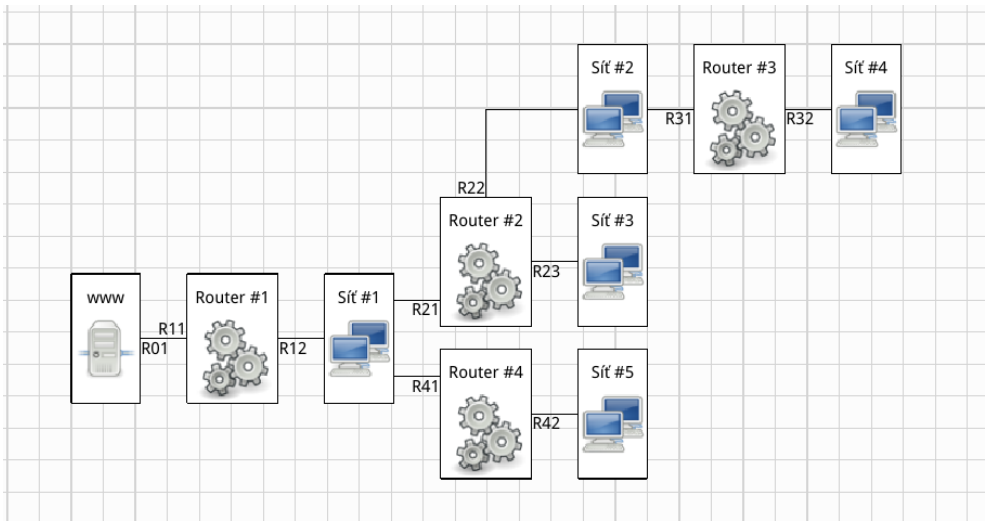
K nastavení vykreslované struktury slouží metoda `drawGraph`, která má jeden argument typu `RoutingTest *`. `RoutingTest` je třída, která generuje strukturu sítě. Tato třída je popsána v kapitole 2.2.2. Metoda `drawGraph` získá ze třídy `RoutingTest` informace o hranách vedoucích ze sítí do routerů a hranách vedoucích z routerů do sítí. Tyto informace jsou reprezentovány vektory dvojic, kde jeden člen dvojice vždy obsahuje číslo routeru a druhý



Obrázek 2.2: Tvorba widgetu pomocí nástroje Qt Creator

člen číslo sítě, nebo obráceně. Samotný graf sítě je vykreslen pouze pomocí těchto informací.

Možný příklad vykreslené topologie znázorňuje obrázek 2.3.



Obrázek 2.3: Ukázka vykreslené topologie pomocí knihovny QGV

2.1.4 Směrovací tabulka

Směrovací tabulka je tvořena třídou `RoutingTable`. Tato třída přepíná pomocí widgetu `QStackedWidget` mezi dvěma zobrazovanými widgety. První widget je vstupní tabulka, která pro kadý router zobrazuje dvojici sloupců síť a next hop. Druhý widget je výstupní tabulka, která zobrazuje výsledky po vyhodnocení testu. Oba tyto widgety jsou ukázány pod sebou na obrázku 2.4.

Třída `RoutingTable` využívá třídu `RoutingTest`, popsanou v sekci 2.2.2, k získání správných výsledků a tyto výsledky pouze porovná s uživatelským vstupem. Třída obsahuje metodu `setOptions`, která v argumentech přijímá nastavení testu a na základě těchto nastavení vygeneruje pomocí `RoutingTest` novou strukturu sítě. Tu zobrazí pomocí volání metody `drawGraph` třídy `NetworkImage`.

Router #1		Router #2		Router #3	
Síť	Next Hop	Síť	Next Hop	Síť	Next Hop
2	R21				
3	R21				
default	R01				

Router #1		Router #2		Router #3	
Síť	Next Hop	Síť	Next Hop	Síť	Next Hop
2	R21	+ 3	R31	+ default	R22
3	R21	+ default	R12		
default	R01				

Obrázek 2.4: Ukázka vstupní tabulky směrování (nahoře) a tabulky zobrazující výsledky (dole)

2.1.5 Tabulka adresace

Tabulka adresace je tvořena třídou `AddressTable`, která funguje podobně jako předchozí třída `RoutingTable`. `AddressTable` také přepíná mezi vstupím widgetem a widgetem s výsledky. Pro test uživatelského vstupu využívá jednu ze tříd `Validator`, `InputOrder` nebo `SizeOrder` v závislosti na zvoleném kritériu hodnocení. Tyto třídy jsou popsány v sekci 2.2.1. Třída `AddressTable` má také metodu `setOptions`, která nastaví minimum a maximum počtu stanic, typ adresy, počet sítí a kritérium hodnocení. Volání této metody vždy vygeneruje nové zadání.

Vstupní widget a widget s výsledky jsou zobrazeny pod sebou na obrázku 2.5.

Síť	Počet	Adresa	Maska	První	Poslední	Broadcast
Síť #1	423	12.0.0.0	23	12.0.0.1	12.0.1.254	12.0.1.255
Síť #2	325					
Síť #3	317					

Síť	Počet	Adresa	Maska	První	Poslední	Broadcast
Síť #1	423	12.0.0.0	23	12.0.0.1	12.0.1.254	12.0.1.255
		12.0.0.0	**23**	**12.0.0.1**	**12.0.1.254**	**12.0.1.255**
Síť #2	325					
		12.0.2.0	**23**	**12.0.2.1**	**12.0.3.254**	**12.0.3.255**

Obrázek 2.5: Ukázka vstupní tabulky adresace (nahoře) a tabulky zobrazující výsledky (dole)

2.2 Jádro

2.2.1 Test adresace

Testování uživatelského vstupu z tabulky adresace má na starost třída `AddressTest` a od ní odvozené třídy `Validator`, `SizeOrder` a `InputOrder`, které testují vstup podle různých kritérií. Třída `SizeOrder` testuje vstup podle velikosti sítí, třída `InputOrder` podle pořadí sítí a třída `Validator` přijímá jakékoliv správné řešení. Tato kritéria jsou blíže popsána v sekci 1.5.2.1 v předchozí kapitole. Třída `AddressTest` obsahuje abstraktní metodu `add`, která slouží pro vložení sítě. Výsledky testu si lze projít pro každou síť zvlášť pomocí iterátoru, který vrací metoda `begin`. Metoda `end` slouží k ověření konce iterace.

Třída `AddressTest` a její podtřídy jsou vlastně šablony, jejichž parametrem je třída reprezentující typ sítí, které budou vkládány. Tímto typem může být buď třída `IPv4Network`, nebo `IPv6Network`.

`IPv4Network` a `IPv6Network` jsou odvozené od abstraktní třídy `Network`. Tyto třídy reprezentují vkládané sítě a obsahují metody, které umožňují třídám odvozeným ze třídy `AddressTest` provádět testy jednotlivých sítí. Třídy `IPv6Network` a `IPv4Network` umožní například otestovat, zda spolu adresní rozsahy dvou sítí vzájemně nekolidují, nebo dokáží pro jednu síť vygenerovat další síť, která na ni bude adresním rozsahem navazovat s minimální možnou mezerou.

K reprezentování IP adres slouží třídy `IPv4` a `IPv6` odvozené od abstraktní třídy `IPAddr`.

2.2.2 Test směrování

Generování struktury topologie má na starost třída `RoutingTest`. Tato třída podle zadaného počtu sítí a routerů generuje náhodnou strukturu sítě s ohle-

dem na pravidla popsaná v sekci 1.5.1 v předchozí kapitole.

Rozhraní třídy tvoří tři metody. Metoda `getResult` vrací vektor, kde každý prvek tohoto vektoru tvoří správné řešení pro jednu směrovací tabulku. Hlavička této metody a struktura `Result`, reprezentující správné řešení, je vidět v následující ukázce:

```
vector<Result> getResult(void) const;

struct RoutingTest::Result {
   vector<pair<int, int>> table;
};
```

Obsah struktury `Result` je pouze jeden vektor dvojic, kde první prvek dvojice určuje číslo sítě a druhý prvek číslo rozhraní následujícího routeru na cestě k této síti.

Další dvě metody rozhraní jsou metody `REdges` a `NEdges`. Tyto metody využívá GUI třída `NetvorkImage` k získání informací o uzlech a hranách mezi nimi. Jejich hlavičky jsou opět v následující ukázce:

```
vector<pair<int, int>> REdges(void) const;
vector<pair<int, int>> NEdges(void) const;
```

Metody vracejí vektory dvojic, kde prvky těchto dvojic reprezentují čísla uzlů, mezi kterými hrana vede. Metoda `REdges` vrací hrany směřující od routeru k síti a metoda `NEdges` naopak.

Kapitola 3

Testování

V této kapitole se zaměřím na testování funkčnosti aplikace a to ze dvou pohledů. Nejprve popíši testování aplikace na několika různých operačních systémech, čímž se snažím otestovat splnění požadavku na multiplatformnost. Poté se zaměřím na průběh testování jednotlivých funkcí aplikace. Na závěr této kapitoly uvedu návod k instalaci aplikace na testovaných systémech.

3.1 Testování OS

Jedním z požadavků na mou aplikaci byla multiplatformnost. Vzhledem k tomu, že veškeré nástroje, které jsem použil, jsou multiplatformní, měla by má aplikace fungovat na všech nejpoužívanějších operačních systémech. Abych se přesvědčil, že má aplikace tento požadavek opravdu splňuje, rozhodl jsem se otestovat sestavení zdrojových kódů a spuštění aplikace na třech vybraných systémech. Následující text popisuje průběh a případné problémy při sestavování aplikace na různých systémech, návod k instalaci na všechny tyto systémy jsem popsal v instalační příručce v sekci 3.3 v této kapitole.

3.1.1 Xubuntu

Celý projekt vznikal na 64 bitové verzi systému Xubuntu 14.04 LTS a proto nebylo spuštění aplikace na tomto systému problém. Pro kompilaci a spuštění aplikace na tomto systému bylo nutné nainstalovat Qt framework verze 5.4 a balík graphviz-dev, který lze stáhnout z dostupných repozitářů ve verzi 2.36, s kterou má aplikace bez problému funguje.

3.1.2 Debian

Další systém, na kterém jsem aplikaci testoval, byla 32 bitová verze systému Debian 7. V tomto systému se vyskytly potíže s balíčkem graphviz-dev. Po přepnutí do okna s vykreslenou topologií docházelo pravidelně k pádu aplikace.

Pomocí debuggeru GDB jsem zjistil, že k problému dochází při vykreslování topologie knihovnou QGV již v konstruktoru třídy `QGVScene`, konkrétně při volání funkce `gvContext`. Funkce `gvContext` je funkce knihovny `gvc` z nástroje `Graphviz`. Z výpisu debuggeru bylo vidět, že funkce `gvContext` dálé volá řadu dalších funkcí a k samotnému pádu dochází až v jedné z nich. Více detailů o tom, proč k problému dochází, se mi zjistit nepodařilo, ale bylo jasné, že problém je někde v knihovně nástroje Graphviz. Porovnal jsem tedy verze balíku graphviz-dev na tomto systému s verzí na předchozím systému Xubuntu 14.04. Na systému Xubuntu 14.04 byla verze 2.36, zatímco na Debianu 7 byla nejvyšší dostupná verze, kterou lze stáhnout pomocí nástroje apt-get verze 2.26. Pokusil jsem se tedy stáhnout instalační balíčky poslední verze 2.38 přímo ze stránek projektu `http://www.graphviz.org`. Zde se však vyskytl problém s nesplněnými závislostmi na jiných balíčcích a instalace se nezdařila. Tento problém vznikl z důvodu, že stažené balíčky graphviz-dev verze 2.38 jsou určeny pro novější verzi systému Debian 8.

Nakonec jsem tedy ze stránek Graphvizu stáhnul archiv, obsahující přimo zdrojové kódy projektu ve verzi 2.38. Tento projekt se podařilo bez problému sestavit a tím vyřešit problém s pádem aplikace.

Dále jsem aplikaci otestoval po upgradu systému na poslední verzi Debian 8. Zde již bylo možné stáhnout balík graphviz-dev ve verzi 2.38 přímo z dostupných repozitářů nástrojem apt-get a instalace i spuštění aplikace proběhlo již bez problému.

3.1.3 Windows 7

Poslední systém, na kterém jsem aplikaci testoval, byl 32 bitový systém Windows 7. Pro sestavení projektu jsem použil nástroj MinGW, který je možno nainstalovat zároveň s instalací Qt frameworku. MinGW je opensource GCC port pro Windows, což znamená, že umožňuje na Windows sestavovat projekty vytvořené pro GCC. MinGW tedy obsahuje C a C++ kompilátor, sadu nástrojů jako například debugger gdb nebo program make pro sestavování projektů, sadu hlavičkových souborů a další užitečné nástroje.

Nástroj Graphviz jsem stáhnul přímo ze stránek projektu. Ty pro Windows nabízí pouze verzi 2.38, a to jako zip soubor, nebo msi soubor.

Celý projekt jsem se nejprve pokoušel sestavit pomocí zvlášť nainstalovaného MinGW verze 4.8. Tato verze však měla problém s překladem některých funkcí z C++11, jako například funkce `to_string`, kterou jsem ve svém projektu využíval. Proto jsem pomocí Qt instalátoru doinstaloval MinGW verzi 4.9, s kterou se již celý projekt podařilo bez problému sestavit.

3.2 Testování funkčnosti

Fungování celé aplikace lze rozdělit na dvě části. První část, odpovědnou za testování adresace a druhou část, odpovědnou za generování topologie a test

směrování. Způsob testování těchto částí popíši zvláš v následujícím textu. Se samotným testováním jsem požádal o pomoc několik kolegů.

3.2.1 Adresace

Pro otestování správné funkčnosti části jádra odpovědné za testování adresace jsem vytvořil nový zdrojový soubor. Tento soubor vytváří kompletní spustitelný program, který čte ze standardního vstupu popis sítí a poté vypisuje sítě, které jsou zadány chybně. Pokud je vstup správný, program nic nevypisuje. Vstup do programu může být například následující:

```
5
331 331 489 15 489

23 12.0.0.0 12.0.0.1 12.0.1.254 12.0.1.255
23 12.0.2.0 12.0.2.1 12.0.3.254 12.0.3.255
23 12.0.4.0 12.0.4.1 12.0.5.254 12.0.5.255
27 12.0.6.0 12.0.6.1 12.0.6.30 12.0.6.31
23 12.0.8.0 12.0.8.1 12.0.9.254 12.0.9.255
```

Na prvním řádku je číslo udávající počet podsítí a za ním následuje seznam hodnot určující velikosti jednotlivých podsítí. Poté následuje popis těchto sítí, kde první hodnota je maska sítě, za ní následuje adresa sítě, adresa první stanice v síti, adresa poslední stanice v síti a pro IPv4 sítě i broadcast adresa.

To, jaké kritérium nebo jakou verzi IP adresy bude program testovat, lze ovlivnit pomocí direktivy `#define` na začátku zdrojového souboru. Například `#define IPv4` znamená, že bude použit IPv4 typ adres a `#define Validator` znamená, že jako kritérium bude použito libovolné validní řešení. Zdrojový kód programu je součástí projektu na přiloženém CD.

Tento program jsem vytvořil, aby bylo možné testovat třídu AddresTest a její podtřídy pro konkrétní velikosti sítí, což ve výsledné aplikaci nelze, ta umožňuje pouze volit rozmezí pro náhodně generované velikosti a proto není příliš vhodná pro testování funkčnosti. Díky tomu lze například jako testovací data využít výstup z aplikace `https://dsn.felk.cvut.cz/projects/psitest/`, kterou je tato práce inspirována a u kterých lze předpokládat, že budou v pořádku.

3.2.2 Směrování

Test směrování probíhá tak, že se pouze porovnává vstup od uživatele se správným řešením. Proto jsem tuto funkcionalitu testoval pouze vyhodnocováním nevyplněných testů a poté jsem kontroloval, zda jsou údaje, které systém zobrazil jako správné řešení, opravdu správně. Vzhledem k tomu, že ve struktuře vygenerované topologie se z pohledu směrování opakují stále stejné vzory, tak se domnívám, že je tato kontrola dostačující a že nemůže dojít při náhodném

generování síťových topologií k vygenerování nějaké neobvyklé struktury, při které by došlo k chybě.

3.3 Instalační příručka

3.3.1 Instalace Qt frameworku

Pro spuštění aplikace je nutné mít v systému nainstalovanou knihovnu Qt. Aplikace byla testována na verzi 5.4, ale měla by stačit libovolná verze alespoň od verze 5.0.

Opensource verzi Qt lze stáhnout ze stránek `http://www.qt.io/download-open-source/`. Zde si lze vybrat buď online instalaci, kde se veškeré potřebné součásti stáhnou až během instalace, nebo offline instalaci, která již veškeré potřebné součásti obsahuje. Stažený program je instalační průvodce, který nainstaluje potřebné knihovny, hlavičkové soubory a nástroje jako Qt Creator, nebo qmake. Ve Windows průvodce umožní navíc nainstalovat nástroj MinGW.

3.3.1.1 Linux

V Linuxu je nejprve nutné nastavit práva spustitelnosti na stažený soubor, například pomocí:

```
chmod +x qt-opensource-linux-x64-1.6.0-8-online.run
```

a poté spustit

```
./qt-opensource-linux-x64-1.6.0-8-online.run
```

Dále je potřeba doinstalovat několik chybějících modulů a knihoven, například pomocí příkazu apt-get [9]:

```
sudo apt-get install mesa-common-dev freeglut3 freeglut3-dev
 libglu1-mesa-dev
sudo apt-get install glut3 glu3-devlibglut3 libglut3-dev
sudo apt-get install libglib2.0-dev libSM-dev libxrender-dev
 libfontconfig1-dev libxext-dev
sudo apt-get install install libgl-dev libglu-dev
```

3.3.1.2 Windows

V systému Windows je třeba po instalaci nastavit cesty k dll souborům, které se nachází v adresáři s nainstalovaným Qt frameworkem, například po instalaci Qt do adresáře `C:`
`Qt`:

```
xpath path %path%;C:\Qt\5.4\mingw491_32\bin
xpath path %path%;C:\Qt\Tools\mingw491_32\bin
```

Konkrétní cesty se mohou lišit podle verze Qt frameworku a MinGW překladače.

3.3.2 Instalace nástroje Graphviz

Aplikace dále vyžaduje knihovny a hlavičkové soubory z nástroje Graphviz, které je možné stáhnout ze stránek `www.graphviz.org/Download.php`. Úspěšně otestované jsou verze 2.36 a 2.38, s verzí 2.26 se vyskytly potíže, které jsou popsané v sekci 3.1.2 v této kapitole.

3.3.2.1 Linux

V Linuxu je možné nainstalovat balíček graphviz-dev přímo z repozitářů, například pomocí příkazu apt-get

```
sudo apt-get install graphviz-dev
```

Nebo pomocí libovolného správce balíků. Další možností je stáhnout balíčky nebo přímo zdrojové kódy z uvedených stránek projektu.

3.3.2.2 Windows

V systému Windows stačí z uvedených stránek stáhnout zip soubor a rozbalit jej do libovolného adresáře. Poté je třeba nastavit cestu k dll knihovnám v tomto adresáři pomocí příkazu

```
xpath path %path%;<cesta k adresáři>/release/bin
```

3.3.3 Sestavení projektu a spuštění

K úspěšnému sestavení projektu je třeba nainstalovat kompilátor g++ a nástroj make. V Linuxu například pomoci příkazu apt-get

```
sudo apt-get install g++ make
```

Ve Windows lze nainstalovat nástroj MinGW obsahující kompilátor i nástroj make při instalaci Qt frameworku.

Dále je ve Windows potřeba nastavit v adresáři s projektem v souboru src/QGVCore/GraphViz.pri proměnnou `GRAPHVIZ_PATH`, do které je třeba uložit cestu k adresáři s nástrojem Graphviz, například pokud je adresář umístěn přímo v `C:/`:

```
GRAPHVIZ_PATH = "C:/graphviz/release"
```

Součástí zdrojových kódů projektu je také projektový soubor BP.pro, který usnadní sestavení celého projektu. Stačí tedy pouze v příkazovém řádku v adresáři se souborem BP.pro spustit program qmake, který se nachází v adresáři s nainstalovaným Qt frameworkem. Tento program vygeneruje soubor Makefile do stejného adresáře. Poté stačí jen ve stejném adresáři spustit program make, který celý projekt sestaví.

V Linuxu tedy stačí v terminálu v adresáři s projektem napsat například:

```
~/Qt/5.4/gcc_64/bin/qmake
make
```

A ve Windows:

```
qmake
mingw32-make
```

Aplikace je poté zkompilována a umístěna v adresáři bin. Ve Windows jde již přímo spustit, v Linuxu je třeba ještě nastavit cestu ke knihovnám libQGV, které se nachází v adresáři lib. To lze udělat pomocí příkazu:

```
export LD_LIBRARY_PATH="$LD_LIBRARY_PATH:<adresář s projektem>/lib"
```

3.3.4 Instalace z balíčku

Na Linuxu je další možností nainstalovat aplikaci z deb balíčků přiložených na CD v adresáři packages. V adresáři se nachází dva balíčky, jeden pro 64 bit systém a druhý pro 32 bit systém. Pro splnění závislostí je nejprve nutné nainstalovat balíčky `qt5-default` a `graphviz-dev`. To je možné např. pomocí příkazu:

```
sudo apt-get install qt5-default graphviz-dev
```

Poté lze nainstalovat aplikaci pomocí příkazu:

```
sudo dpkg -i <zvolený balíček>
```

Nyní je již balíček nainstalován a aplikace lze spustit příkazem `addrtest`.

Závěr

V této práci se mi podařilo implementovat funkční nástroj sloužící ke generování a vyhodnocování testů adresace a směrování v počítačových sítích. Podařilo se mi rozšířit možnosti již existujícího řešení používaného v předmětu BI-PSI, ze kterého tato práce vychází.

Součástí práce je i rozbor a výběr nástroje pro tvorbu GUI, kde jsem vybíral mezi třemi knihovnami: Tcl/Tk, GTK+ a Qt. Po zvážení jednotlivých možností jsem vybral nástroj Qt.

Dále jsem se snažil najít vhodnou knihovnu, která by umožnila vykreslit topologii do okna Qt aplikace. Podařilo se mi nalézt knihovnu QGV, která téměř odpovídala mým požadavkům, ale pro použití bylo třeba provést několik úprav. Nejpodstatnější úpravou bylo rozšíření tohoto nástroje o možnost zarovnání popisků hran na začátek nebo konec hrany. Původně knihovna QGV umožňovala zarovnání popisku pouze na střed.

Jelikož jedním z požadavků byla multiplatformnost, byla výsledná aplikace nakonec otestována na linuxových systémech Debian a Xubuntu a na systému Windows. Na všech těchto systémech se podařilo aplikaci sestavit a poté i spustit.

Literatura

[1] Postel, J.: Internet protocol. RFC 791, RFC Editor, September 1981. Dostupné z: http://www.rfc-editor.org/rfc/rfc791.txt

[2] Postel, J.; Mogul, J.: Internet Standard Subnetting Procedure. RFC 950, RFC Editor, August 1985. Dostupné z: http://www.rfc-editor.org/rfc/rfc950.txt

[3] Hinden, R.; Deering, S.: IP Version 6 Addressing Architecture. RFC 4291, RFC Editor, February 2006. Dostupné z: http://www.rfc-editor.org/rfc/rfc4291.txt

[4] Baker, F.: Requirements for IP Version 4 Routers. RFC 1812, RFC Editor, June 1995. Dostupné z: http://www.rfc-editor.org/rfc/rfc1812.txt

[5] Tcl community: Tcl Developer Xchange [online]. October 2005, [Cited 2015-05-07]. Dostupné z: http://www.tcl.tk

[6] The GTK+ Team: The GTK+ Project [online]. ©2007 - 2015, [Cited 2015-05-07]. Dostupné z: http://www.gtk.org

[7] C++/Tk: A complete C++ interface to the Tk GUI toolkit [online]. [Cited 2015-05-07]. Dostupné z: http://cpptk.sourceforge.net/

[8] The Qt Company: *Qt Documentation [online]*. ©2015, [Cited 2015-05-07]. Dostupné z: http://doc.qt.io

[9] Chtoboczek, M.: *Grafická uživatelská rozhraní v Qt a C++*. Brno: Computer Press, první vydání, 2013, ISBN 978-80-251-4124-3.

[10] TU Dortmund: *Open Graph Drawing Framework [online]*. ©1999 - 2012, [Cited 2015-05-07]. Dostupné z: http://igraph.org/c/doc

[11] The igraph core team: *igraph Reference Manual [online]*. ©2003 - 2015, [Cited 2015-05-07]. Dostupné z: http://igraph.org

[12] Gansner, E. R.: *Using Graphviz as a Library [online]*. August 2014, [Cited 2015-05-07]. Dostupné z: http://www.graphviz.org/doc/libguide/libguide.pdf

[13] Tým vyvíjející knihovnu Qanava: Qanava: Qt Canvas Avancé – Open Source Graphic library for Qt [online]. [Cited 2015-05-07]. Dostupné z: http://www.qanava.org/

[14] QGV: Interactive Qt graphViz display [software]. 2014, [Cited 2015-05-07]. Dostupné z: https://github.com/zipoo256/qgv

Příloha **A**

Seznam použitých zkratek

IP Internet Protocol

IPv4 Internet Protocol version 4

IPv6 Internet Protocol version 6

GUI Graphical user interface

XML Extensible markup language

GTK GIMP Toolkit

OGDF Open Graph Drawing Framework

QGV Qt graphViz

DOT graph description language

MSVC Microsoft Visual C++

GDB GNU Debugger

GCC GNU Compiler Collection

Příloha **B**

Obsah přiloženého CD

```
readme.txt ..................................... stručný popis obsahu CD
├── packages ................... adresář s instalačními .deb balíčky pro Linux
│   ├── addrtest_0.1_amd64.deb ..................... verze pro 64-bit systém
│   └── addrtest_0.1_i386.deb ...................... verze pro 32-bit systém
├── project
│   ├── doc ................................... programátorská dokumentace
│   ├── src ................................... zdrojové kódy implementace
│   ├── BP.pro ........................................ projekový soubor
│   ├── Doxyfile ......................... konfigurační soubor pro Doxygen
│   ├── readme.txt ..................................... návod k instalaci
│   └── project.tar.gz ............................ archiv adresáře project
└── thesis ..................................................... text práce
    ├── Latex ........... adresář se zdrojovou formou práce ve formátu LaTeX
    ├── BP_Kužel_Jakub_2015.pdf ............... text práce ve formátu PDF
    └── zadani.pdf ..................................... zadání této práce
```