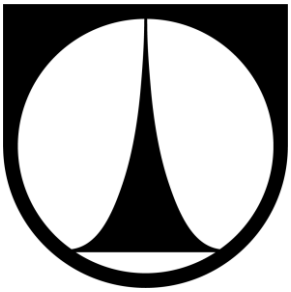
TECHNICKÁ UNIVERZITA V LIBERCI

FAKULTA MECHATRONIKY, INFORMATIKY A MEZIOBOROVÝCH STUDIÍ

Ústav nových technologií a aplikované informatiky



NÁSTROJ PRO VIZUALIZACI SÍŤOVÉ KOMUNIKACE

TOOL FOR VISUALIZATION OF NETWORK COMMUNICATION

DIPLOMOVÁ PRÁCE

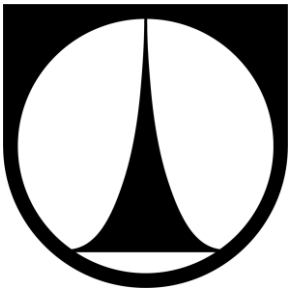
Bc. Vít Dittrich

Květen 2013

TECHNICKÁ UNIVERZITA V LIBERCI

FAKULTA MECHATRONIKY, INFORMATIKY A MEZIOBOROVÝCH STUDIÍ

Ústav nových technologií a aplikované informatiky



Studijní program

N2612 Elektrotechnika a informatika

Obor Informační technologie

NÁSTROJ PRO VIZUALIZACI SÍŤOVÉ KOMUNIKACE

TOOL FOR VISUALIZATION OF NETWORK COMMUNICATION

Diplomová práce

Bc. Vít Dittrich

Vedoucí diplomové práce: doc. RNDr. Pavel Satrapa, Ph.D.

Počet stran: 58

Počet obrázků: 14

Počet tabulek: 1

Počet příloh: 2

Květen 2013

TECHNICKÁ UNIVERZITA V LIBERCI
Fakulta mechatroniky, informatiky a mezioborových studií
Akademický rok: **2012/2013**

ZADÁNÍ DIPLOMOVÉ PRÁCE

(PROJEKTU, UMĚLECKÉHO DÍLA, UMĚLECKÉHO VÝKONU)

Jméno a příjmení:	**Bc. Vít Dittrich**
Osobní číslo:	**M11000232**
Studijní program:	**N2612 Elektrotechnika a informatika**
Studijní obor:	**Informační technologie**
Název tématu:	**Nástroj pro vizualizaci síťové komunikace**
Zadávající katedra:	**Ústav nových technologií a aplikované informatiky**

Zásady pro vypracování:

1. Proveďte rešerši existujících nástrojů pro vizualizaci síťové komunikace.
2. Navrhněte webovou aplikaci znázorňující proces směrování a průchodu paketů sítí.
3. Základní požadavky na aplikaci: Možnost definice a úpravy topologie sítě, zadání statických směrovacích tabulek jednotlivých prvků, zadání odesilatele a adresáta paketu, vizualizace směrovacích rozhodnutí a průchodu paketu sítí.
4. Navrženou aplikaci implementujte a otestujte v prostředí nejrozšířenějších webových prohlížečů.

Rozsah grafických prací: **dle potřeby**

Rozsah pracovní zprávy: **60 stran**

Forma zpracování diplomové práce: **tištěná/elektronická**

Seznam odborné literatury:

[1] STEVENS, Richard. TCP/IP iIllustrated. Vyd. 1. Boston: Addison-Wesley, 1994, 576 s. ISBN 02-016-3346-9.
[2] ZELDMAN, Jeffrey. Tvorba webů podle standardů. Vyd. 1. Brno: Computer Press, 2004, 410 s. ISBN 80-251-0347-1.
[3] CASTAGNETTO, Jesus. PHP. Programujeme profesionálně. 1.vyd. Praha: Computer Press, 2001, 656 s. cd. ISBN 80-722-6310-2.
[4] ZAKAS, Nicholas C. JavaScript pro webové vývojáře: programujeme profesionálně. Vyd. 1. Překlad Lukáš Krejčí. Brno: Computer Press, 2009, 832 s. ISBN 978-80-251-2509-0.

Vedoucí diplomové práce: **doc. RNDr. Pavel Satrapa, Ph.D.**
Ústav nových technologií a aplikované informatiky

Datum zadání diplomové práce: **18. října 2012**
Termín odevzdání diplomové práce: **17. května 2013**

prof. Ing. Václav Kopecký, CSc.
děkan

L.S.

prof. Dr. Ing. Jiří Maryška, CSc.
vedoucí ústavu

Téma

Nástroj pro vizualizaci síťové komunikace

Anotace

Diplomová práce se zabývá problematikou síťové komunikace a vizualizace procesu průchodu IP paketu počítačovou sítí. Pozornost je věnována zhodnocení stávajících aplikací, shrnutí možností při tvorbě aplikace, výběru vhodného řešení, teorii počítačových sítí a vlastnímu vývoji aplikace. Důraz je kladen na vhodný způsob zobrazení výběru ze směrovacích tabulek jednotlivých prvků a celkové koncepce aplikace tak, aby byla vhodná pro síťové začátečníky. V aplikaci byla implementována podpora vizualizace průchodu paketu počítačovou sítí jak pro IPv4 tak pro IPv6 pakety.

Cílem této práce je implementace všech navržených vlastností do ucelené webové aplikace použitelné pro výuku základů počítačových sítí a pro objasnění procesu průchodu paketu počítačovou sítí.

Klíčová slova: IP, vizualizace, počítačová síť, směrování, webová aplikace

Theme

Tool for visualization of network communication

Annotation

This dissertation pursues problematic of network communication and visualization of passing an IP packet through computer network. Attention is dedicated to evaluation of current applications, summarizing options for creating an application, choosing the pertinent solution, theory of computer networks and development of an own application. Emphasis is put on an appropriate method of displaying the selection from routing tables of particular components and holistic concept of the application so that it is suitable for network beginners. In this application, there has been implemented the support for visualization of passing the packet through computer network for both IPv4 and IPv6 packets.

The aim of this work is to implement all proposed attributes to a comprehensive web application, which could be used for schooling the basics of computer networks and to demonstrate passing the packet through computer network.

Key words: IP, visualization, computer network, routing, web application

Zpracovatel: Technická Univerzita v Liberci, Fakulta mechatroniky, informatiky a mezioborových studií, Ústav nových technologií a aplikované informatiky

Dokončeno: 2013

Prohlášení

Byl jsem seznámen s tím, že na mou diplomovou práci se plně vztahuje zákon č. 121/2000 Sb. o právu autorském, zejména § 60 – školní dílo.

Beru na vědomí, že Technická univerzita v Liberci (TUL) nezasahuje do mých autorských práv užitím mé diplomové práce pro vnitřní potřebu TUL.

Užiji-li diplomovou práci nebo poskytnu-li licenci k jejímu využití, jsem si vědom povinnosti informovat o této skutečnosti TUL; v tomto případě má TUL právo ode mne požadovat úhradu nákladů, které vynaložila na vytvoření díla, až do jejich skutečné výše.

Diplomovou práci jsem vypracoval samostatně s použitím uvedené literatury a na základě konzultací s vedoucím diplomové práce.

V …… dne …….. ……

podpis

Poděkování

Na tomto místě bych rád poděkoval vedoucímu diplomové práce doc. RNDr. Pavlu Satrapovi, Ph.D. za odbornou pomoc, rady a konzultace.

Dále bych rád poděkoval především rodičům a všem mým blízkým, kteří mě po celou dobu studia na FM TUL podporovali a byli mi po dobu studia pevnou oporou.

Obsah

Seznam obrázků

Seznam tabulek

Přehled použitých zkratek, symbolů

AJAX	Asynchrounous JavaScript and XML
API	Application Programming Interface
ARP	Address Resolution Protocol
BGP	Border Gateway Protocol
CIDR	Classless Inter-Domain Routing
CLI	Command-Line Interface
CSMA	Carrier Sense Multiple Access
CSS	Cascading Style Sheet
DBMS	Database Management System
DNS	Domain Name System
DTD	Data Type Definition
ECMA	European Computer Manufacturers Association
FTP	File Transfer Protocol
HTML	HyperText Markup Language
IEEE	Institute of Electrical and Electronics Engineers
IP	Internet Protocol
IPv4	Internet Protocol version 4
IPv6	Internet Protocol version 6
IS-IS	Intermediate System to Intermediate System
ISO/OSI	International Organization for Standardization/Open Systems Interconnection
IT	Information Technology
ITU	Internet Telecommunications Union
JSON	JavaScript Object Notation
LAN	Local Area Network
MAC	Media Access Control
MTU	Maximum Transmission Unit
NAT	Network Address Translation
OSPF	Open Shortest Path First
RFC	Request For Comments

PHP	Hypertext Preprocessor
RIP	Routing Information Protocol
SSH	Secure Shell
SQL	Structured Query Language
TCP	Transmission Control Protocol
TTL	Time To Live
URI	Uniform Resourcce Identifier
WAN	Wide Area Network
W3C	World Wide Web Consorcium
XML	Extensible Markup Language

1. Úvod

Podíváme-li se do historie vývoje počítačových sítí, ať už se jedná o firemní nebo osobní využití, množství a velikost počítačových sítí neustále roste. S příchodem mobilních zařízení a implementací síťových rozhraní do nejrůznějších zařízení v domácnostech i průmyslových odvětvích také přichází větší nároky na administrátory počítačových sítí. Se stále rostoucím počtem připojených zařízení vyvstává také množství stále sofistikovanějších protokolů ať již spojové nebo transportní vrstvy. Základy však zůstávají takřka neměnné a bez nich je velmi obtížné až téměř nemožné pochopit další návaznosti v počítačových sítích.

Velká část dnešní společnosti si například síť internet, jež je největší počítačovou sítí světa, představuje jako jakousi „černou krabičku“, která začíná a končí na konci kabelu, kterým je připojen jejich domácí směrovač. Stejně tak řada začínajících síťových administrátorů nebo studentů IT oborů si pak nedovede představit, jak na pozadí počítačová síť principiálně pracuje. Bohužel existuje velice málo vizualizačních nástrojů, které by dokázaly jednoduchým a srozumitelným způsobem, bez více či méně pokročilých znalostí tuto situaci zlepšit. Záleží tak na představivosti konkrétního studenta a tím i rychlosti a schopnosti dále pokračovat ve studiu a pochopení sofistikovanějších protokolů, které na elementárních znalostech počítačových sítí staví.

Existující řešení pro vizualizaci průchodu paketu počítačovou sítí většinou vyžadují pokročilé znalosti konfigurace síťových prvků a jsou zpravidla orientovány na zařízení konkrétního výrobce. V těchto případech je software pro vizualizace zpravidla zpoplatněn nebo zpřístupněn pouze uživatelům nějakým způsobem napojeným na tyto výrobce, a je zaměřen na simulaci chování konkrétní definované sítě. Další podstatná část nástrojů sloužící k vizualizaci je pak uniplatformní nebo dokáže zobrazit pouze výslednou trasu požadovaného spojení.

1.1. Cíle diplomové práce

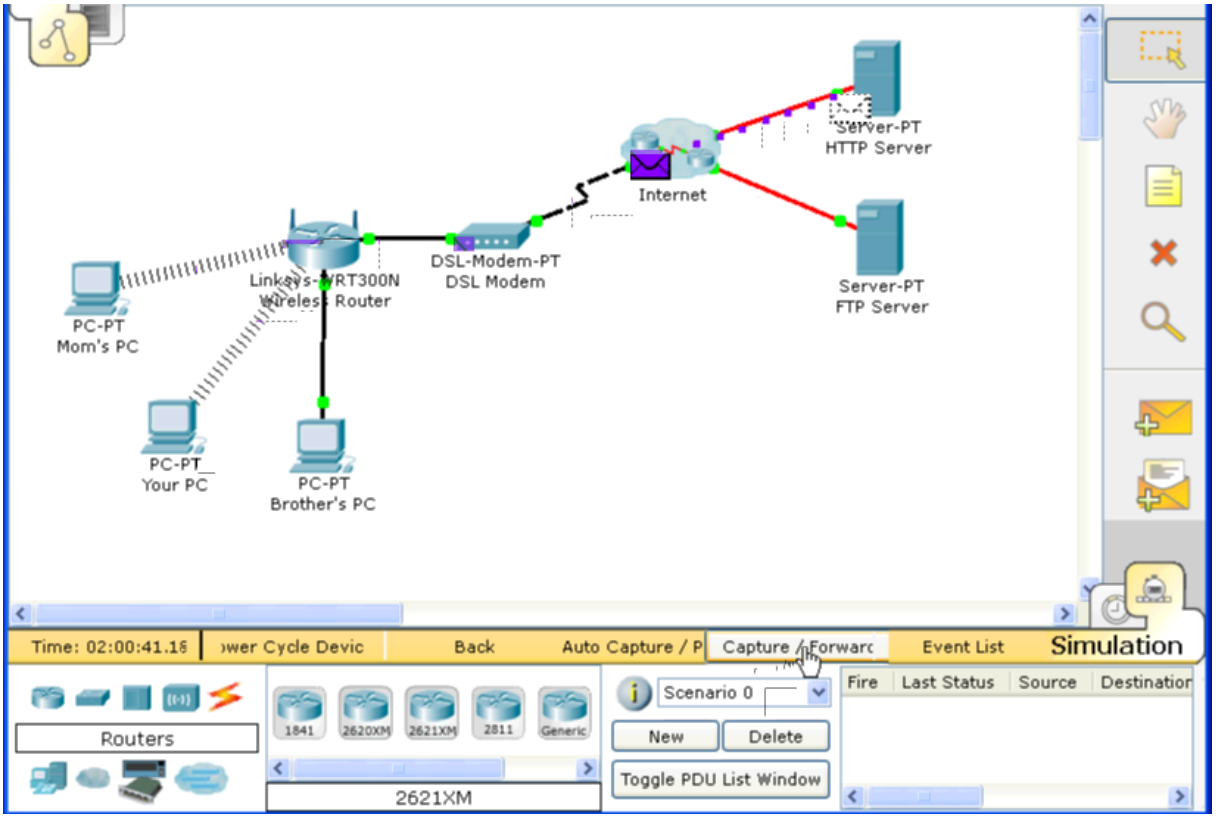
V této diplomové práci budu řešit návrh a tvorbu webové aplikace sloužící k vizualizaci průchodu paketu počítačovou sítí, kde je kladen důraz na zobrazení rozhodování ve směrovacích tabulkách jednotlivých prvků.

V práci se nejdříve budu zabývat rešerší stávajících řešení. Následně pak možností tvorby webových aplikací a výběrem vhodného řešení. Dále budu rozebírat reálné fungování počítačových sítí a zjednodušení pro potřeby vyvíjené aplikace. V poslední části pak bude popsána konkrétní implementace webové aplikace a její testování v nejrozšířenějších webových prohlížečích.

2. Rešerše stávajících řešení

2.1. Cisco Packet Tracer

Cisco Packet Tracer je software pro simulaci chování počítačové sítě postavené na aktivních prvcích společnosti Cisco Systems, Inc. Tento program byl společností Cisco vyvíjen pouze pro operační systém Microsoft Windows a následně byl taktéž od verze 5.3 portován i pro Linuxové distribuce. Primární využití je v rámci výuky v Cisco Networking Academy. Pro účastníky, absolventy a lektory Cisco Networking Academy je tento software poskytován zdarma.



Obrázek 2.1: Cisco Packet Tracer [1]

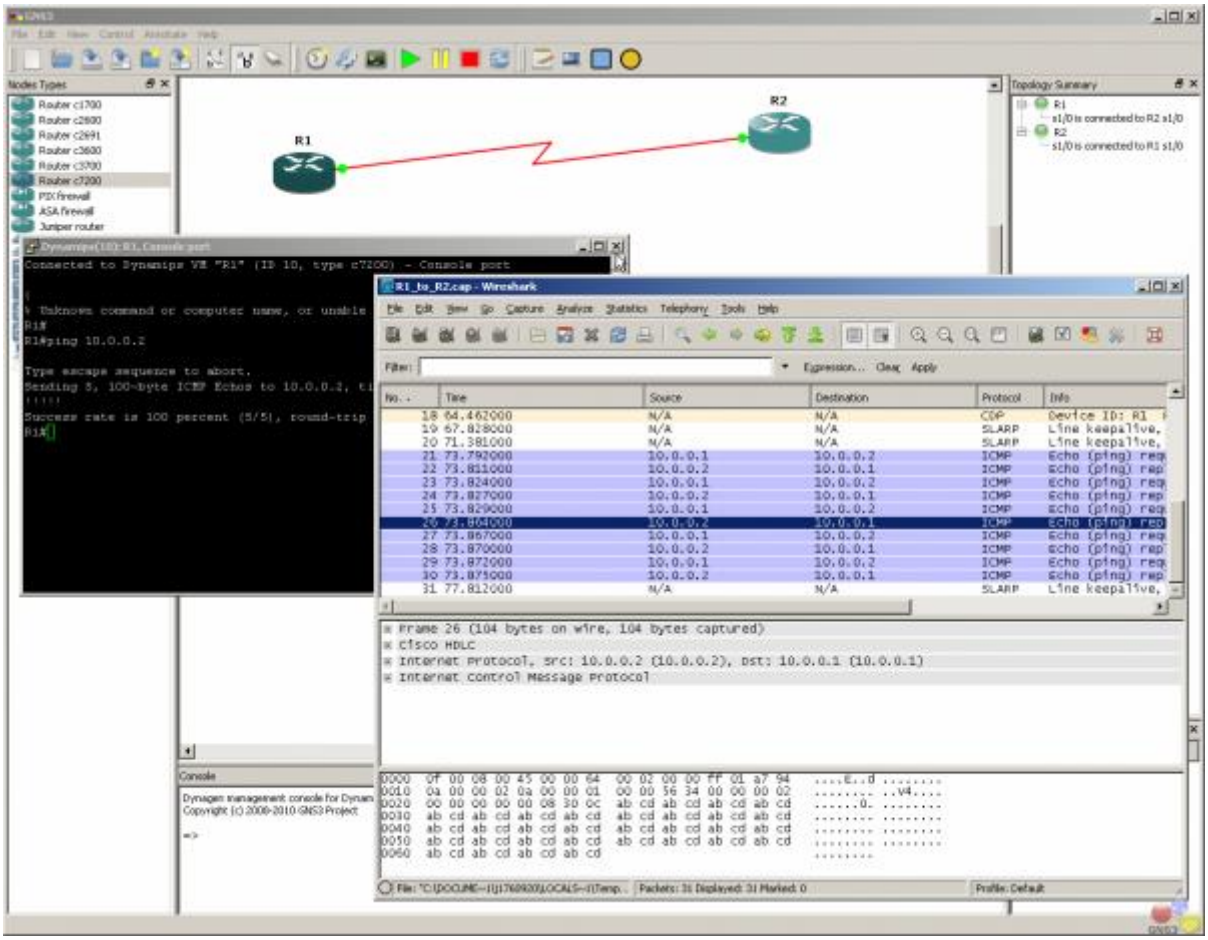
V programu Cisco Packet Tracer lze virtuálně vytvořit téměř libovolnou síť z komponent společnosti Cisco Systems a následně pak simulovat jakýkoliv provoz touto sítí. V rámci diagnostiky sítě je také možné „odchytávání“ jednotlivých rámců a paketů v síti. Velmi dobře propracované jsou směrovače a přepínače společnosti Cisco. Ty se funkčností i ovládáním neliší od reálných prvků.

Vzhledem k rozsahu výuky v rámci Cisco Networking Academy, která se zaměřuje především na síťové administrátory, a používání tohoto software je v rámci výuky vysvětleno, je zaměřen spíše na pokročilé uživatele počítačových sítí a nikoliv na začátečníky. Jak je již

z předchozího textu patrné, je pro využívání software nutná znalost konfigurace jednotlivých síťových prvků společnosti Cisco Systems pomocí CLI. Vzhledem k těmto skutečnostem je tak software cílen na jinou kategorii uživatelů, než na kterou se zaměřuje tato diplomová práce.

2.2. Graphical Network Simulator

Graphical Network Simulator je program, který je určen síťovým specialistům a administrátorům sítí pro vytváření a následnou simulaci virtuálních počítačových sítí. Aplikace umožnuje virtuální nakonfigurování jednotlivých síťových prvků (směrovačů, přepínačů, a koncových stanic). Následně je možné simulovat virtuální síťový provoz včetně jeho analýzy.



Obrázek 2.2: Graphical Network Simulator

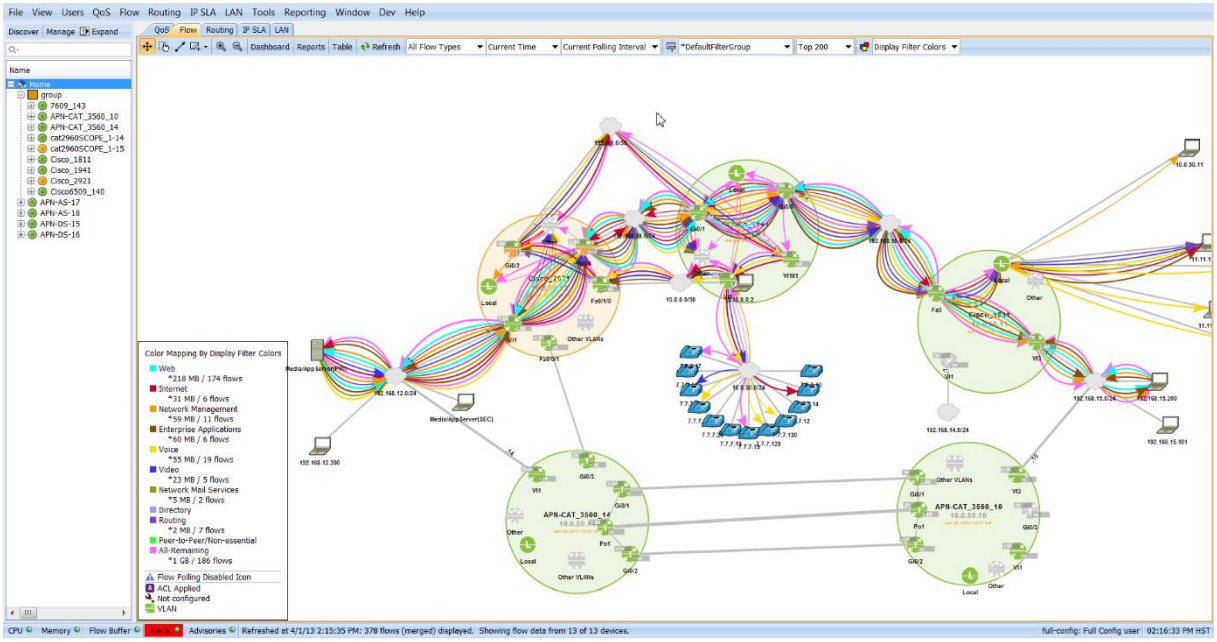
Na rozdíl od Cisco Packet Traceru je Graphical Network Simulator distribuován pod licencí GPL a je tedy volně k dispozici. I z tohoto důvodu se jedná o multiplatformní řešení, které je dostupné jak pro operační systémy z rodiny Microsoft Windows, tak pro UNIXové operační systémy. Na webových stránkách projektu [4] je taktéž možné stáhnout předinstalované

řešení virtuálního operačního systému pro multiplatformní virtualizační nástroj Oracle VM VirtualBox. To lze považovat za značnou výhodu, jelikož odpadá nutnost konfigurace software a lze se tak vyvarovat případných problémů s nekompatibilitou.

Oproti Cisco Packet Traceru je v Graphical Network Simulator možné konfigurovat nejen prvky společnosti Cisco Systems, ale například i prvky společnosti Juniper Networks a dalších výrobců. Zde je velikou výhodou otevřenost zdrojového kódu. Aplikace je nativní a je nutné ji instalovat. Není tedy možné tuto aplikaci provozovat na nestandartních zařízeních, jako jsou například tablety. Stejně jako Cisco Packet Tracer je aplikace určena především pokročilým uživatelům a administrátorům sítí, nikoliv začátečníkům. Obecně tak lze říci, že se jedná o opensourcovou alternativu k programu Cisco Packet Tracer s rozšířením o možnost konfigurace prvků jiných výrobců.

2.3. LiveAction

Dalším porovnávaným řešením je software LiveAction společnosti ActionPacked! Networks. Tento software slouží pro vizualizaci topologie, datových toků, směrování a dalších aktivit počítačových sítí. Nejedná se zde přitom o pouhou simulaci. Aplikace může být napojena na reálnou síť (preferovány jsou prvky společnosti Cisco Systems) a tato síť následně může být virtuálně testována a následné změny mohou být na reálnou síť aplikovány.



Obrázek 2.3: LiveAction - vizualizace toků

Aplikace slouží také jako management a monitoring sítě. V aplikaci tak lze s již velmi pokročilými znalostmi počítačových sítí simulovat datové toky (nikoliv průchod jednotlivých paketů) a vyhodnocovat tyto informace. Aplikace dokáže tvořit ucelené a kvalitní výstupy o chování reálné sítě. I simulace je zde poměrně vizuálně strohá, protože při manipulaci s aplikací jsou předpokládány pokročilé znalosti.

Aplikace je nativní, placená a určená pro počítačové stanice s operačním systémem Microsoft Windows. Spíše než vizualizační nástroj slouží aplikace primárně pro monitoring a management sítí – ačkoliv jisté vizualizace a simulace umožňuje, nelze ji proto jednoduše srovnávat s ostatním softwarem primárně určeným pro oblast vizualizace.

2.4. IP Routing Simulation

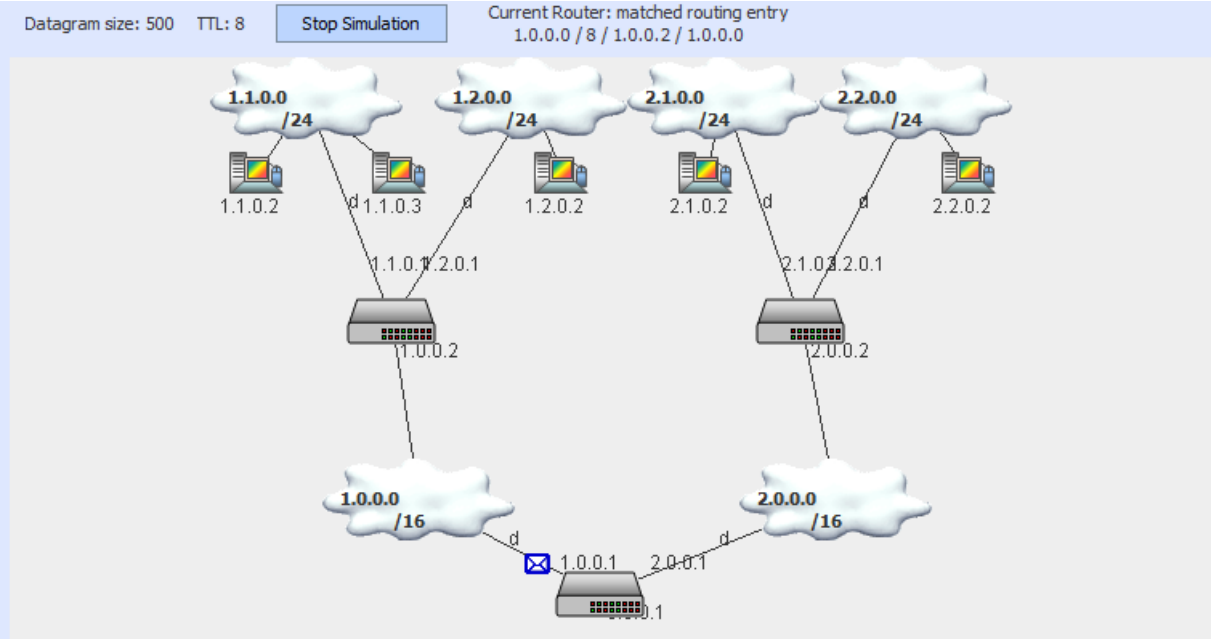
IP Routing Simulation je aplikace napsaná jako webový applet v programovacím jazyce Java na univerzitě v Tel Avivu [7]. Jedná se o poměrně zdařilý simulátor průchodu paketu počítačovou sítí, který umožňuje vytvořit jednoduchou virtuální síť a následně do této sítě definovaný paket vypustit.

Applet umožňuje nadefinovat jednotlivé sítě a do nich následně připojit směrovače a koncové stanice, kterým je přiřazena adresa z dané sítě. Dále je možné nadefinovat na každém směrovači směrovací tabulku.

Po připojení koncových stanic do sítě je možné vložit a editovat paket, kterému můžeme vybrat odesilatele a adresáta. Dále je zde možné definovat velikost MTU a hodnotu TTL. Poté můžeme paket sítí odeslat. Při průchodu paketu sítí je znázorněna cesta paketu, vypsána velikost MTU a hodnota TTL. Při každém průchodu směrovačem je taktéž vypsána nejlepší shoda ve směrovací tabulce. Simulace lze taktéž zastavit. Simulace je však přerušena a nelze v simulaci dále pokračovat - je potřeba ji spustit znovu.

K aplikaci je poměrně propracovaný anglicky psaný manuál. I přesto je aplikace poměrně špatně ovladatelná. Přidávání některých parametrů není intuitivní a i pokročilý uživatel musí občas využít k aplikaci manuál. Rozhraní jednotlivých zařízení nelze editovat ani manuálně přidávat, což může být v některých situacích limitující. Další nevýhodou je nemožnost pozastavení aplikace. Začátečník nemusí některé parametry postřehnout a nemá čas se nad nimi zamýšlet. V případě chyby je pak obecně vypsána pouze informace, že se

simulace



Obrázek 2.4: IP Routing Simulation [7]

nezdařila. Důvod - například vypršení hodnoty TTL - již uveden není. Velice dobře je naopak zpracovaná definice paketu.

Celkově tak aplikace splňuje základní požadavek na simulaci průchodu paketu počítačovou sítí, vizualizace rozhodování ve směrovacích tabulkách však nenabízí. Dále zde není zapracována možnost ukládání vytvořené mapy a podpora IPv6. Vytknout lze i těžkopádné ovládání aplikace. V poslední řadě je nevýhodou, že aplikace pracuje jako applet a je tedy nutné mít doinstalovaný dodatečný software, který není možné na některé platformy doinstalovat, nebo je potřeba pro uživatele dodatečných oprávnění.

2.5. Shrnutí stávajících řešení

Stávající aplikace se zaměřují spíše na simulace a vizualizace pokročilých vlastností počítačových sítí. Většina těchto aplikací je pak psána jako nativní. Pro výuku základů IP směrování je pak většina těchto řešení příliš náročná.

Nejblíže zadání diplomové práci je svým řešením software IP Routing Simulation, kde se jedná o webovou aplikaci a umožňuje jednoduchou vizualizaci průchodu počítačovou IP sítí. Bohužel však nevizualizuje výběr ve směrovací tabulce, nepodporuje směrování protokolu IPv6 a tím, že je použit Java applet je aplikace závislá na podpoře Javy. Z tohoto důvodu je vhodné vytvoření vlastní webové aplikace pro vizualizaci průchodu paketu počítačovou sítí.

3. Možnosti realizace

Webové grafické rozhraní lze v moderních prohlížečích realizovat různými způsoby. V případě této diplomové práce byly zvažovány tři varianty, které byly hodnoceny z hlediska kompatibility s webovými prohlížeči a s tím spojenými problémy na straně uživatelů i aplikace samotné.

3.1. Applet

První zvažovanou možností bylo vytvoření aplikace jako Java Appletu. Applet je softwarová komponenta, která běží v kontextu jiného programu – v tomto případě Java Virtual Machine. Applet je většinou orientován na plnění konkrétní funkce v daném kontextu. Značnou výhodou je nezávislost na serverové straně a poměrně vysoký výkon, který ovšem udává výkon samotné stanice hosta.

Nevýhodou tohoto řešení je nutnost instalace dodatečného software na stanici uživatele, což nemusí být vždy možné například z důvodu uživatelských práv nebo nekompatibility dané platformy (mobilní zařízení,…). V případě využití jako učební pomůcky není toto řešení příliš vhodné, jelikož vyžaduje dodatečná oprávnění nebo zkušenosti na straně uživatele.

Další nevýhodou může být zranitelnost uživatelského zařízení prostřednictvím Java appletu. S tím souvisí i blokování Java appletů ze strany správců uživatelských stanic ve firmách nebo některých školních sítích.

3.2. Canvas a HTML5

HTML 5 je v současnosti stále ve fázi návrhu organizací W3C, proto i implementace podpory do prohlížečů není v mnoha případech zcela korektní a je potřeba k tomuto přihlédnout. Element <canvas> v HTML 5 slouží k vykreslování grafiky (většinou pomocí JavaScriptu). Jedná se pouze o kontejner, ve kterém je grafika vykreslována. U elementu canvas lze dále definovat šířku kreslicího „plátna“, výšku a například ID. Lze tak mít v jedné stránce více kreslicích ploch.

Velkou výhodou elementu canvas je jeho určení. Protože je přímo navržen pro vykreslování grafiky, je zde umožněno velmi jednoduše vykreslovat základní i pokročilé tvary 2D grafiky. Některé prohlížeče již většinou experimentálně podporují i vykreslování prvků 3D grafiky.

Práce s grafickými prvky v elementu canvas je poměrně jednoduchá a například pro vykreslení přímky postačuje následující kód, ve kterém stačí definovat počáteční a koncový bod přímky.

```
<script type="text/javascript">
var c=document.getElementById("myCanvas");
var ctx=c.getContext("2d");
ctx.moveTo(10,10);
ctx.lineTo(150,50);
ctx.stroke();
</script>
```

Jistou nevýhodou jinak velmi vhodného elementu canvas je jeho podpora ve webových prohlížečích. Například mainstreamový prohlížeč Internet Explorer jej podporuje až od verze 9. K březnu 2013 téměř 11,5% uživatelů internetu v ČR využívá pro přístup k webovému obsahu starší verze prohlížeče Internet Explorer [12]. Ačkoliv podíl uživatelů, kteří používají prohlížeče nepodporující element canvas, za poslední rok klesl o více než 4%, není toto procento uživatelů zanedbatelné. Z veřejně dostupných informací společnosti SPIR z. s. p. o. provozující portál NetMonitor.cz [8] bylo v únoru 2013 celkem více jak šest a půl milionu českých internetových uživatelů, z čehož dále plyne, že cca 700 tisíc těchto uživatelů tak využívá prohlížeč nepodporující element canvas.

Vzhledem k nejednotné implementaci a stále nezanedbatelnému množství uživatelů využívajících prohlížeče nepodporující element canvas by nebylo vhodné využít tohoto řešení.

3.3. JavaScript

JavaScript je objektově orientovaný multiplatformní skriptovací jazyk používaný jako interpretovaný programovací jazyk pro webové aplikace. Program v JavaScriptu se obvykle spouští až po stažení webové stránky ze serveru na straně klienta. Na rozdíl od jiných interpretovaných programovacích jazyků, například PHP, které se spouštějí na straně serveru ještě před jejich stažením stránky. Vzhledem ke stáří a neměnnosti základů JavaScriptu (byl standardizován společností ECMA již v roce 1997) je i poměrně široká podpora implementace na straně prohlížečů.

I v této oblasti však panují jisté výjimky. Některé události v různých prohlížečích mohou generovat různé výsledky, případně se jedna událost v různých prohlížečích generuje jinak. Příkladem může být metoda target, kterou podporuje většina prohlížečů, avšak v Internet Exploreru je využívána metoda srcElement pro získání stejných dat. Naštěstí je valná většina

těchto výjimek poměrně zdařile zdokumentována a není proto problém se při vývoji aplikace vyhnout chybám.

Pro JavaScript byla vyvinuta i řada knihoven, které jsou většinou programovány se znalostí odlišností různých prohlížečů, a proto lze využít těchto knihoven pro značné ulehčení řešení tohoto rozličného chování. Občasné problémy pak nastávají pouze při využívání velmi starých verzí webových prohlížečů. Mezi nejznámější knihovny patří například jQuery, MooTools, Dojo a další.

3.4. Výběr vhodného řešení

Vzhledem k velmi dobré podpoře napříč webovými prohlížeči na nejrůznějších platformách se po základním srovnání jako vhodné řešení pro tuto diplomovou práci jeví kombinace JavaScriptu, knihovny jQuery, technologie AJAX a backendového řešení postaveného na PHP a databázovém systému MySQL. Toto řešení je vhodné především z důvodu nároků na uživatele, který nemusí instalovat dodatečný software, a také lze toto řešení využít na nejrůznějších platformách od nejnovějších televizorů s webovým prohlížečem, přes mobilní telefony, po klasické počítačové stanice s nejrůznějším operačním systémem.

5. Teorie počítačových sítí

Pro úspěšné vytvoření funkční aplikace je nutné správně pochopit a implementovat důležité mechanismy počítačových sítí založených na protokolu IP. Pro potřeby vizualizačního nástroje je podstatné určit, které mechanizmy do aplikace implementovat a které zjednodušit.

5.1. Modely ISO/OSI a TCP/IP

Pro nejobecnější popis počítačových sítí se používá standardizovaný referenční model ISO/OSI, který standardizovala organizace ISO již v roce 1984 jako mezinárodní normu ISO 7498. Tento referenční model se používá jako názorný příklad řešení síťové komunikace pomocí vrstveného modelu, kde jsou jednotlivé vrstvy nezávislé.

Model ISO/OSI nespecifikuje konkrétní implementaci jednotlivých vrstev, pouze teoreticky popisuje účel a vlastnosti jednotlivých vrstev. Lze tedy obecně tento model považovat za abstraktní.

ISO/OSI			TCP/IP	
7.	Aplikační vrstva		Aplikační vrstva	4.
6.	Prezenční vrstva			
5.	Relační vrstva			
4.	Transportní vrstva		Transportní vrstva	3.
3.	Síťová vrstva		Síťová vrstva (IP)	2.
2.	Spojová vrstva		Vrstva síťového rozhraní	1
1.	Fyzická vrstva			

Obrázek 5.1: Model ISO/OSI a TCP/IP

Referenční model ISO/OSI je sedmivrstvý. Každá ze sedmi vrstev vykonává skupinu jasně definovaných funkcí potřebných pro komunikaci. Pro svou činnost pak využívá služeb sousední nižší vrstvy a naopak své služby poskytuje vrstvě vyšší. Referenční model nedovoluje vrstvy vynechávat, avšak některá z vrstev nemusí být nutně aktivní – ta je pak označována jako transparentní.

Jednotlivé vrstvy modelu jsou patrné z obrázku 5.1. Fyzická vrstva modelu specifikuje fyzickou komunikaci a definuje všechny elektrické a fyzikální vlastnosti zařízení, které se

o první vrstvu stará. Patří sem například typ konektorů, rozložení pinů, napěťové úrovně, vlastnosti kabelů, způsob přenosu bitů, způsob modulace, kódování a další. Mezi známé zástupce fyzické vrstvy patří například RS-232. Definice fyzické vrstvy je také součástí specifikací IEEE 802.3 (Ethernet) nebo 802.11 (Wi-Fi).

Spojová vrstva poskytuje spojení mezi dvěma sousedními systémy. Uspořádává data fyzické vrstvy do rámců – logických celků. Spojová vrstva zajišťuje přístup ke sdílenému médiu (například pomocí metod CSMA) a adresaci na fyzickém spojení v jednom síťovém segmentu. K adresaci jsou obvykle použity 48bitové fyzické (MAC) adresy. Dále má spojová vrstva na starosti detekci a případně opravu přenosových chyb. Data spojové vrstvy jsou opatřena fyzickou adresou. Příkladem zástupce spojové vrstvy může být obecně IEEE 802. Na druhé vrstvě pracují síťové mosty a přepínače.

Síťová vrstva se stará o směrování a adresování v jednotlivých sítích. Poskytuje spojení mezi systémy, které spolu přímo nesousedí, a to díky funkcím, které dokáží překlenout rozdílné vlastnosti jednotlivých přenosových sítí. Tato vrstva poskytuje funkce pro zajištění přenosu dat různé délky od zdroje k příjemci skrze jednu nebo několik vzájemně propojených sítí. Vrstva dále poskytuje směrovací funkce a informuje o problémech při doručování dat. Jednotkou informace je na třetí vrstvě paket. Mezi nejznámější zástupce třetí vrstvy patří Internetový Protokol a například protokol X.25. Na třetí vrstvě pracují síťové směrovače.

Transportní vrstva zajišťuje přenos dat mezi koncovými uzly. Účelem vrstvy je poskytnout takovou kvalitu přenosu, kterou požadují vyšší vrstvy. Mezi hlavní zástupce transportní vrstvy patří protokol UDP (nespojově orientovaný), který zajišťuje přenos bez záruk, ale i zbytečného prodlení. Dalším zástupcem transportní vrstvy je protokol TCP (spojově orientovaný), který zajišťuje přenos se zárukami, kdy se snaží dosáhnout stavu, kdy data jsou spolehlivě doručena přesně v té podobě, ve které byla odeslána.

Relační vrstva by měla organizovat a synchronizovat dialog mezi spolupracujícími relačními vrstvami obou komunikujících systémů a řídit mezi nimi výměnu dat. Umožnuje zahájení, ukončení synchronizace, obnoveni relačního spojení a oznamování výjimečných stavů. K jednotlivým paketům přiřazuje synchronizační značky, které může využít k poskládání původního pořadí dat.

Prezenční vrstva má transformovat data do tvaru, který požadují jednotlivé aplikace. Data se navíc transformují pro účel přenosu skrze nižší vrstvy. Prezenční vrstva se zabývá pouze

strukturou dat – například přizpůsobí pořadí jednotlivých bajtů – ale již se nezajímá o jejich význam, o což se stará vrstva vyšší.

Aplikační vrstva poskytuje konkrétním aplikacím přístup ke komunikačnímu systému a umožňuje tak jejich spolupráci napříč počítačovou sítí. Do této vrstvy by spadaly například služby a protokoly jako DNS, FTP, SSH a další.

Model TCP/IP je taktéž modelem vrstveným. Jak vyplývá z obrázku 5.1, model TCP/IP je jakýmsi zjednodušením referenčního modelu ISO/OSI. Nejedná se již o model teoretický, ale reálný. Tento model zjednodušuje původních sedm vrstev na vrstvy čtyři. Spojová a fyzická vrstva se spojuje na vrstvu síťového rozhraní. Vrstvy aplikační, prezenční a relační pak spojuje do vrstvy aplikační.

Vrstva síťového rozhraní má tak na starosti vše, co je spojeno s ovládáním konkrétní přenosové sítě a s přímým vysíláním a příjmem datových rámců. V rámci TCP/IP není tato vrstva konkrétně specifikována, protože je přímo závislá na použité technologii.

Aplikační vrstva napřímo komunikuje s vrstvou transportní. Tato vrstva však musí zastat funkci všech tří vrstev z modelu ISO/OSI. Zajišťuje tak případné prezenční a relační služby. Na rozdíl od modelu ISO/OSI si tak jednotlivé aplikace tyto potřeby musí realizovat samy.

5.2. Protokol IP a adresace

Internet Protokol je jedním z nejzákladnějších protokolů pracujících na síťové vrstvě. Protokol sám o sobě neposkytuje záruky na přenos dat a tuto službu ponechává vyšším vrstvám. IP protokol je zodpovědný za směrování paketů ze zdrojového zařízení na zařízení cílové, přes jednu nebo více IP sítí. Paket se skládá z řídících dat (hlavičky) a uživatelských dat. Hlavička paketu obsahuje zdrojovou a cílovou adresu, hodnotu TTL, kontrolní součty informací a může obsahovat další data. Důležité je zmínit, že doručení dat v IP síti není zaručeno, stejně jako není zaručeno pořadí, v jakém pakety k cíli dorazí, ani zda nebudou zduplikovány. V IP sítích se tak data mohou ztrácet nebo pokaždé putovat jinou cestou.

V současné době se rozlišuje na Internet Protocol verze 4 a Internet Protocol verze 6, které se především liší délkou adresy, která u starší verze 4 využívá 32 bitů, kdežto u novější verze 6 již využívá pro adresaci 128 bitů. Tyto protokoly nejsou mezi sebou kompatibilní. Sítě jsou tak provozovány jako takzvaný „dual stack“. Z větší délky adresy plyne i velikost adresního

prostoru, která je u IPv4 teoreticky 2^{32}, což představuje cca 4 miliardy adres a u IPv6 pak 2^{128}, což je cca 340 sextiliónů (340×10^{36}). Dalších vylepšení doznal protokol například v pevné délce hlavičky každého paketu, nebo zdokonalení schopnosti přenášet větší množství dat, zabudování šifrování do síťové vrstvy a mnoho dalších vylepšení. Základní funkce a práce s protokolem IP však zůstala až na drobné změny stejná.

Každé síťové rozhraní v jakémkoli síťovém zařízení může mít přiřazenu unikátní IP adresu. Zpravidla tak u IPv4 má přiřazenu právě jednu adresu, ale může mít adres přiřazeno více nebo naopak žádnou. Pokud například síťový most nemá přiřazenou žádnou IP adresu, na jeho funkčnosti to nic nemění. IP adresa však nemusí být v každém případě unikátní – například adresa použitá jako anycastová může být propagována z více zařízení v síti, nebo v případě privátních rozsahů a použití technologie NAT u IPv4 tak může být konkrétní adresa použita v celosvětovém měřítku opakovaně. V každém případě by však v rámci jedné podsítě neměla (pro správnou funkčnost sítě nesmí) být jedna IP adresa použita vícekrát.

Zápis IPv4 adres je ve tvaru čtyř čísel v rozmezí 0-255 oddělených tečkami. IPv4 adresa tak například může vypadat jako 147.230.16.27, což je adresa webového serveru Technické Univerzity v Liberci. IPv6 adresy jsou zapisovány jako 8 čtyřmístných hexadecimálních číslic oddělených dvojtečkami. Série po sobě jdoucích nul může být nahrazena dvojtečkami a nuly z leva se mohou vynechávat. Adresa tak ekvivalentně může být zapisována například v následujících tvarech, kdy se stále jedná o stejnou IPv6 adresu.

```
2001:07f8:0014:0000:0000:00000:0001:00002
2001:07f8:0014::00000:0001:00002
2001:07f8:0014::0001:00002
2001:7f8:14::1:2
```

Každá IP adresa spadá do určité podsítě. Podsíť, určená maskou sítě, je definovaná část sítě. Maska sítě zapsaná v binárním tvaru má zleva samé jedničky až do místa, kde končí číslo sítě a dále je doplněna nulami na místech pro číslo síťového rozraní. Pomocí masky sítě pak směrovače rozhodují o směrování IP paketů sítí. Maska sítě udává rozsah dané podsítě – tedy její velikost.

V IPv4 se maska hojně zapisovala ve stejném tvaru jako IPv4 adresa – tedy například 255.255.255.0, což je binárně 24 jedniček. Postupně (a především s rozvojem IPv6 adres, kde takovýto zápis není vhodný) se přechází na takzvaný CIDR zápis a tedy /24, což je právě onen

počet jedniček z binárního zápisu výše uvedené masky podsítě, jak je patrné z následujícího zápisu: 11111111.11111111.11111111.00000000

U IPv4 je první adresa z rozsahu adresou sítě, která by měla být v rámci sítě jedinečná. Poslední adresou z rozsahu u IPv4 adres je takzvaná broadcastová adresa, tedy adresa určená pro vyslání paketu všem zařízením připojeným v dané podsíti. U IPv6 se těchto mechanismů nepoužívá. Namísto broadcastové adresy plní tuto funkci multicast, který je – na rozdíl od IPv4 – přímo zahrnut ve specifikaci.

5.3. Směrování

Každé dvě sítě, které spolu - přímo či nepřímo - sousedí, musí být pro vzájemnou komunikaci spojeny pomocí směrovačů. Každý směrovač obsahuje směrovací tabulku. Jedná se o datovou strukturu uloženou v operační paměti směrovače, která slouží k výběru nejlepší cesty pro směrování jednotlivých paketů v počítačové síti. Tabulka obsahuje jakousi zjednodušenou topologii sítě, dle které je pak rozhodnutí o směrování prováděno.

Směrovací tabulka je vytvářena při konfiguraci síťového systému. Záznamy ve směrovací tabulce mohou být statické (nastaveny manuálně) nebo dynamické, které se mění podle situace v počítačové síti. Dynamické záznamy přidává a odebírá směrovací obslužný software nějakého sofistikovanějšího směrovacího protokolu (RIP, OSPF, IS-IS, BGP).

Směrovací tabulka obsahuje minimálně adresu sítě, síťovou masku a bránu sítě. Brána je IP adresa následujícího směrovače, kterému mají být pakety předány. Dále může směrovací tabulka obsahovat metriku na základě konkrétního směrovacího protokolu. Metrika je celočíselný údaj, který udává relativní vyjádření ceny při použití dané trasy.

Výběr nejlepšího záznamu ze směrovací tabulky spočívá v určení nejlepší shody s danou sítí. Ve směrovací tabulce tak například můžeme mít záznam pro síť 192.168.0.0/16 a 192.168.1.0/24. V případě, že bude cílovou adresou adresa například 192.168.1.123, vybere se záznam 192.168.1.0/24, protože jde o přesnější určení cílové sítě. V případě, že je pro daný cíl k dispozici více jak jedna trasa, rozhoduje záznam s nižší metrikou. Pro síť označovanou 0.0.0.0/0 nazývanou jako výchozí brána dojde ke shodě vždy, proto je využita ke směrování všech paketů bez implicitního vypsání cíle – například tedy v koncových zařízeních. V IPv6 směrovací tabulce probíhá z hlediska binárního zápisu výběr naprosto totožně.

6. Návrh aplikace

6.1. Analýza

Aplikace má být koncipována jako učební pomůcka pro znázornění topologií sítí a následnou vizualizací průchodu paketu počítačovou sítí. Z tohoto hlediska se jeví jako vhodná možnost ukládání nezávislých síťových schémat, které si bude moci každý uživatel upravovat, exportovat a v budoucnu kdykoliv znovu použít. Z tohoto důvodu bylo při koncepci aplikace zapotřebí umožnit ukládání a správu dat na straně serveru.

Pro potřeby vizualizace a případného budoucího rozšiřování aplikace by měla být definice datových struktur natolik obecná, aby bylo možné kdykoliv aplikaci rozšířit bez potřeb velkých změn stávajícího kódu. Vzhledem k faktu, že by se výsledek aplikace měl co nejvíce přiblížit reálnému schématu sítě, musí návrh struktur aplikace co nejvíce korespondovat s reálným síťovým modelem. Z tohoto pohledu se jeví jako ideální dodržování referenčního TCP/IP.

Pro účely aplikace by pak mělo stačit využít spojovou a síťovou vrstvu. Z hlediska aplikace to pak bude dělení na jednotlivé síťové prvky, jejich rozhraní a spoje mezi těmito rozhraními, IP adresy a směrovací tabulky.

6.2. Návrh aplikace

Webová aplikace funguje ve své podstatě jako aplikace klient-server. Vizuální stránku tvoří klient na základě dat poskytnutých serverem. Server data uchovává, částečně zpracovává a poskytuje klientovi. Serverovou část zde zastupuje webový server se skriptovacím jazykem pro předzpracování dat a databáze pro jejich uložení. Jako klient v tomto případě figuruje webový prohlížeč na straně uživatele.

Pro komunikaci mezi klientem a serverem je potřeba vytvořit rozhraní, které dokáže předat data serveru směrem od uživatele, ale také dokáže poskytnout požadovaná data klientovi. Pro tuto komunikaci je potřeba naprogramovat příslušné API. To musí sloužit pro získávání, ukládání, upravování a mazání dat. Pro tento účel musí sloužit specifické funkce a algoritmy. API musí být taktéž zabezpečeno proti zneužití, aby nebylo možné smazat nebo upravit záznam, který nepatří danému uživateli.

V další fázi bylo potřeba rozvrhnout, která část výpočtů bude ponechána serveru, a která část bude přenechána pro klienta. Klient může být výkonově o poznání slabší nežli server, a proto by složitější výpočty měl obstarat a předpřipravit server. S každým dotazem na server se však přenáší další dodatečná data, což může zatížit pomalejší (například mobilní) připojení a přenos sám o sobě trvá nějaký čas – běžně v řádech milisekund až sekund. V případě výpočtu na straně klienta však může nastat situace, kdy pouze kvůli výpočtu přeneseme neúměrně více dat. Je tedy třeba nalézt kompromis mezi množstvím přenášených dat a složitostí výpočtu pro optimální běh aplikace.

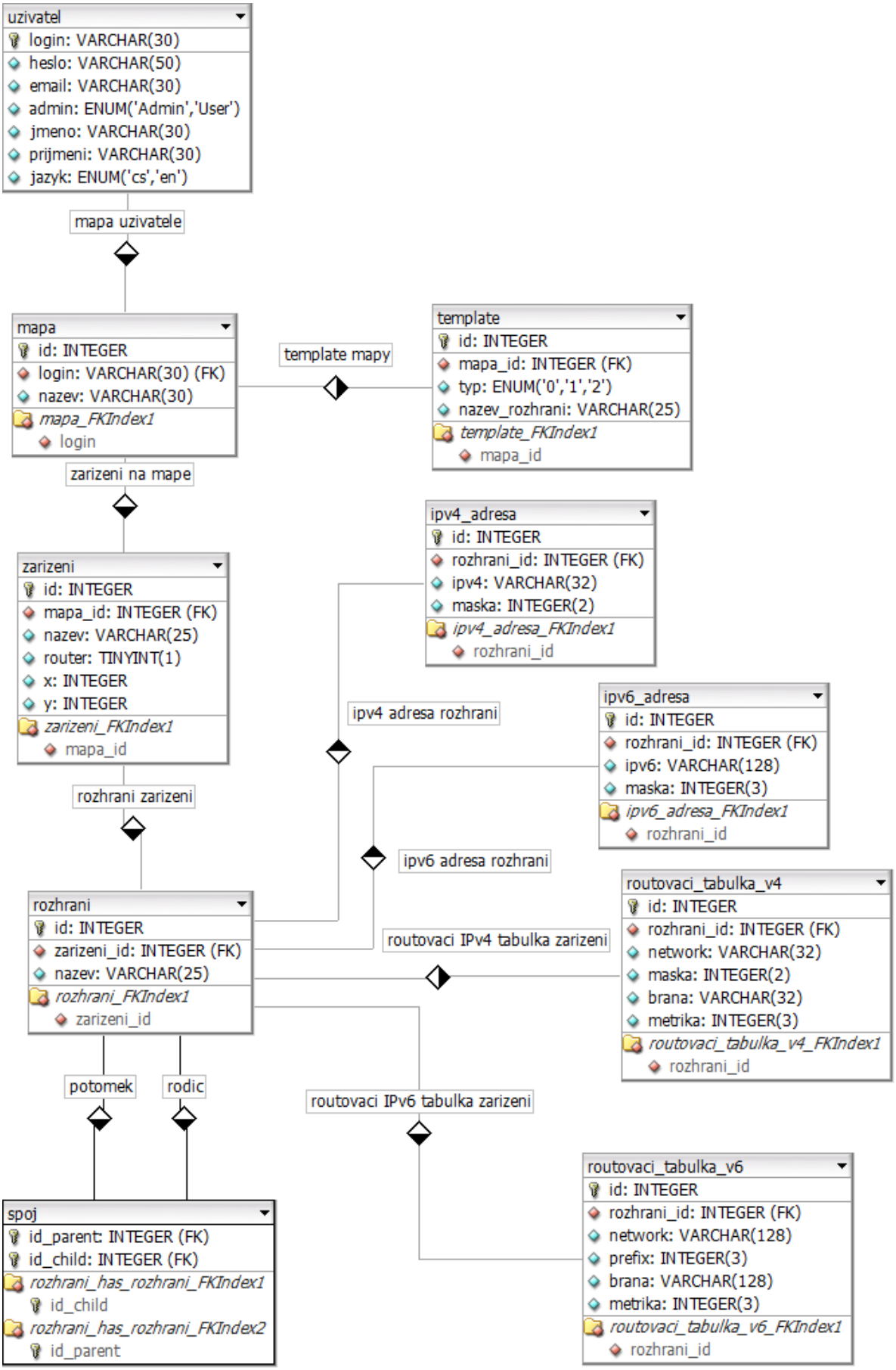
Z hlediska datových formátů je optimální použít pro ukládání IP adres binární zápis, se kterým lze systematičtěji pracovat. Výhodou je pak jednodušší implementace mezi jednotlivými verzemi IP protokolu.

6.3. Návrh databáze

K uložení dat se v aplikaci používá databáze MySQL. Vzhledem ke koncepci aplikace je tedy v prvé řadě důležitý návrh této databáze. Při návrhu databáze bylo postupováno „shora“, tedy vzestupně co se týče modelu TCP/IP. V aplikaci by měl existovat jednoznačně identifikovatelný uživatel. Tento uživatel by měl mít možnost vytvářet svá schémata - mapy. V každé mapě by měla být zařízení, která mají nějaká rozhraní. Tato rozhraní mohou být mezi sebou propojena tak, aby vytvořila spoje. Každé zařízení může mít směrovací tabulku, která obsahuje jednotlivé směrovací záznamy. Každému rozhraní mohou být dále přiřazovány IP adresy.

Celá aplikace je koncipována jako takzvaný „dual stack“. Tedy nezávislé fungování IPv4 a IPv6 sítě. Z tohoto důvodu je tedy nutné i v návrhu databáze oddělit oba protokoly paralelně zvlášť. Z relačního diagramu na obrázku 6.1 je patrné, že obě verze protokolu jsou až na délky adres totožné.

Z relačního diagramu je dále patrné, že každé zařízení má příznak *router*, který slouží ke grafické identifikaci směrovače, přepínače a koncové stanice. Pro grafické uspořádání zde dále slouží souřadnice v ose x a y. Za pozornost stojí taktéž tabulka *template*, která slouží jako šablona pro zařízení v jednotlivých mapách. Každý typ zařízení může mít předdefinované rozhraní a jejich názvy.



Obrázek 6.1: Relační diagram databáze

7. Použité technologie

7.1. AJAX

AJAX je zkratka pro obecné označení technologie pro vývoj interaktivních webových aplikací, které mění jejich obsah nebo předávají serveru další informace „za chodu“ bez nutnosti znovunačtení dat celé stránky. Zkratka AJAX v původním názvu znamená Asynchronous JavaScript and XML. Ačkoliv by se mělo jednat o asynchronní komunikaci pomocí JavaScriptu a XML, jak původní zkratka napovídá, obecně tato technologie nemusí využívat ani JavaScript, ani se data nemusí přenášet pomocí XML. Často je místo XML využíván spíše způsob zápisu (datový formát) JSON.

AJAX je vhodné využít především v interaktivních aplikacích, kde by bylo znovunačtení celého dokumentu nevhodné nebo velmi náročné ať již z hlediska množství přenášených dat nebo náročnosti celkového návrhu aplikace. Ve vhodných případech tak často sníží množství přenášených dat a zvýší interaktivitu výsledné aplikace. Naopak při nevhodném použití může enormně zvýšit počet http dotazů na server, což se může zpětně projevit na výkonu aplikace. Je tedy nutné při návrhu aplikace správně vyhodnotit, která data a ve který okamžik přenášet.

7.2. jQuery

jQuery je malá a rychlá JavaScriptová knihovna, která zjednodušuje práci při interakci mezi JavaScriptem a HTML dokumentem. Mezi hlavní vlastnosti jQuery knihovny patří jednoduché zpracování technologie AJAX, zachytávání vytvářených událostí stránky, nejrůznější možnosti animací a další funkce. Velký důraz je přitom kladen na zpracování všech metod a funkcí tak, aby byla zajištěna podpora ve většině moderních webových prohlížečů.

jQuery většinou existuje jako jeden komprimovaný JavaScriptový soubor, který obsahuje všechny funkce. Tento soubor je pak možné stáhnout ze stránek projektu a následně vložit do vlastní aplikace. Vzhledem k velké oblibě knihovny jQuery a jejímu častému využití ve velkém množství projektů je taktéž možné využít Google AJAX Libraries API nebo načtení ze serverů společnosti Google popřípadě přímo ze stránek projektu. Tento způsob pak nabízí spousty výhod včetně unifikovaného cachování, snížení přenosu dat a tím i snížení odezvy PHP.

7.3. PHP

PHP je skriptovací programovací jazyk určený především pro programování aplikační logiky dynamických webových stránek. Skripty napsané v jazyce PHP jsou prováděny na straně serveru a uživatel vidí pouze výsledný, skriptem vygenerovaný výstup. Zpravidla se jedná o HTML kód. Syntaxe jazyka PHP je inspirována jazyky C nebo Perl. Interpret jazyka PHP je, až na systémové funkce, na platformě nezávislý, a proto i skripty lze bez úprav snadno přenášet.

PHP podporuje celou řadu knihoven určených pro nejrůznější funkce, ať už se jedná o přístupy k různým DBMS nebo o tvorbu grafiky, zpracování textů a další. Vzhledem k rozšířenosti skriptovacího jazyka PHP na webhostingových serverech nebo možnosti jednoduché instalace webového serveru na uživatelskou stanici Windows (například pomocí softwarového balíčku XAMPP), je PHP jednoduchým a kvalitním nástrojem pro vývoj moderních webových aplikací.

7.4. MySQL

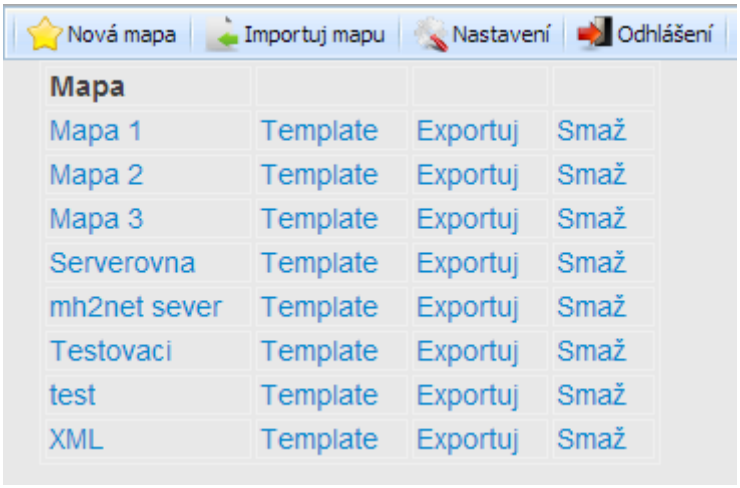
MySQL je jeden z nejrozšířenějších relačních databázových systémů typu DBMS. Jedná se o multiplatformní software s dvojím licencováním – bezplatná verze pod licencí GPL a komerční placená licence. Databáze MySQL je celá optimalizována na rychlost, avšak často i za cenu zjednodušení.

Databáze v MySQL slouží k ukládání dat a skládá se z tabulek. V případě propojení ěchto tabulek jsou pak tyto spojeny na základě klíčů. Komunikace s databází probíhá pomocí standardizovaného dotazovacího jazyka SQL. Podobně jako u ostatních SQL databází se jedná o dialekt tohoto jazyka s některými rozšířeními.

Databáze MySQL je velmi rozšířená díky své jednoduchosti a nenáročnosti. Je vhodná především pro správu a ukládání dat webových aplikací a je často používána v kombinaci se skriptovacím jazykem PHP.

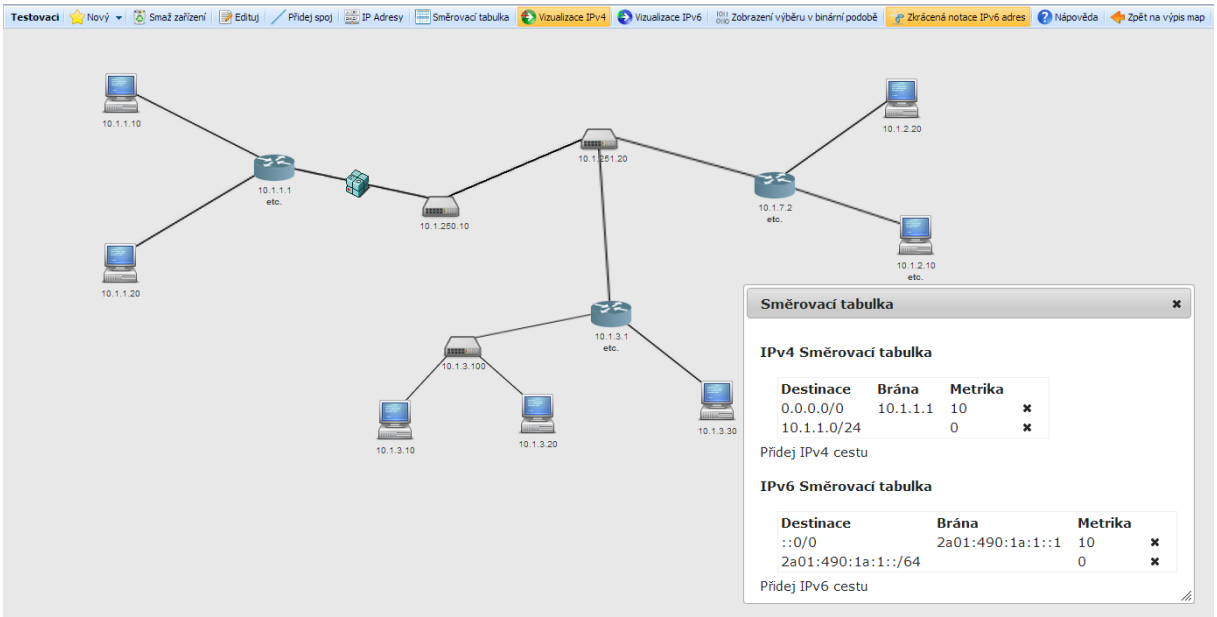
8. Výsledné grafické rozhraní aplikace

Pro snadnější pochopení kontextu následujících kapitol se tato část práce stručně zabývá grafickým rozhraním aplikace. Výsledná aplikace se skládá z několika vzájemně propojených částí. Pro přihlášení je uživatel přesměrován do standardního webového formuláře, kde je nutné vyplnit uživatelské jméno a heslo. Po stisknutí tlačítka odeslat je pak na základě zadaných přihlašovacích údajů uživatel ověřován.



Obrázek 8.1: Hlavní menu

Po přihlášení je uživateli k dispozici jednoduché menu se základními prvky pro import mapy, nastavení, vytvoření nové mapy nebo odhlášení, jak je patrné z obrázku 8.1. U výpisu map je dále možné upravovat jednotlivé šablony, danou mapu exportovat nebo smazat.

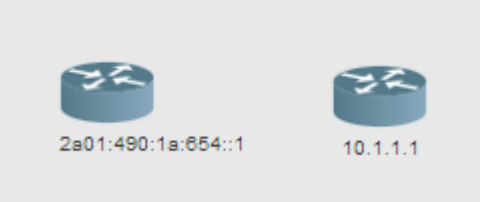


Obrázek 8.2: Ukázka hlavní části aplikace

Při kliknutí na název mapy je uživatel přesměrován do samotné hlavní části aplikace.

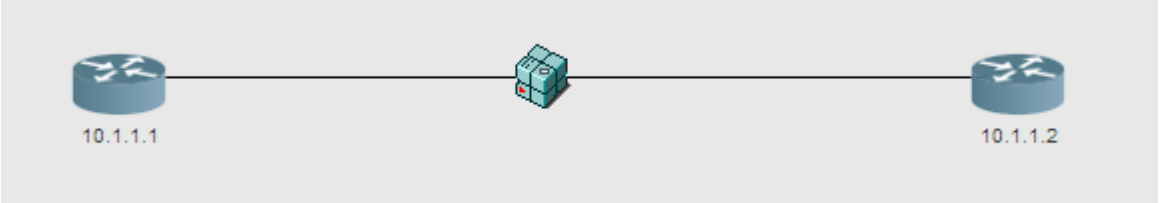
V hlavní části aplikace má uživatel v horní části k dispozici menu pro ovládání základních činností aplikace. Pod ním následuje samotná pracovní plocha, kde jsou vykreslena nejenom všechna zařízení a spoje mezi nimi, ale i jejich popisky. V této části pracovní plochy aplikace jsou též zobrazována dialogová okna pro případnou úpravu jednotlivých prvků. Kompletní náhled na hlavní část aplikace včetně otevřeného dialogového okna směrovací tabulky je zobrazen na obrázku 8.2.

V aplikaci jsou pod jednotlivými prvky zobrazeny první IP adresy. Aplikace při přepínání jednotlivých vizualizací přepíná i příslušné adresy. To je patrné z obrázku 8.3, kde je v levé části zobrazena ukázka IPv6 adresy při vizualizaci IPv6 paketu a v pravé části IPv4 adresa při příslušné vizualizaci IPv4 paketu.



Obrázek 8.3: Ukázka zobrazení IP adres

Po spuštění jednotlivých částí vizualizace je mezi jednotlivými prvky animován obrázek balíčku, který představuje požadovaný paket. Tento obrázek se pohybuje na středu jednotlivých spojů, jak je patrné z obrázku 8.4



Obrázek 8.4: Ukázka paketu při přesunu mezi směrovači

Dalšími důležitými grafickými částmi aplikace jsou například vizualizace směrovacích tabulek nebo dialogová okna. Tyto části jsou podrobněji popsány a zobrazeny v následující kapitole.

9. Implementace

9.1. Vykreslování zařízení a spojů

Základním stavebním prvkem celé aplikace je bezesporu vykreslování. V aplikaci je třeba vykreslovat jednotlivá zařízení a spoje mezi nimi. Jak již bylo zmíněno, JavaScript neobsahuje funkce pro práci s grafikou. Nelze tak tedy jednoduše umisťovat jednotlivá zařízení a vytvářet mezi nimi přímky znázorňující jednotlivé spoje. Tomuto tématu byla věnována pozornost v rámci magisterského projektu [3], na který tato diplomová práce navazuje a v jehož rámci vznikla knihovna funkcí pro vykreslování jednotlivých komponent síťové topologie.

V prvním kroku aplikace bylo potřeba vytvořit potřebné funkce pro přidávání jednotlivých položek do mapy a možnost jejich posunu. Pro přidání slouží funkce *createRouter(id,x,y)*. Obdobně pak slouží funkce *createPC(id,x,y)* a *createSwitch(id,x,y)*. Tyto funkce přidávají vnořený element * * do základního elementu * * s identifikátorem mapa na předem určené souřadnice. V elementu s identifikátorem mapa jsou vykreslována všechna zařízení návrhu. V elementu * * určeném pro zařízení se dále nachází specifický obrázek znázorňující konkrétní zařízení a další element * *, do kterého je následně vypsána IP adresa zařízení.

Následně bylo nutné vytvořit funkce pro vykreslování přímek mezi zadanými body. Na základě článku od francouzského vývojáře webových aplikací Mathieu Henriho [5] bylo tedy třeba tuto funkci vytvořit. Pro toto nám poslouží funkce *drawLine* a *updateLine*.

Funkce fungují na principu „roztahování“ předem určených průhledných obrázků přímek s natočením 45° a 135° v různých velikostech a jednopixelové obrázky pro natočení 0° a 90°. Nejprve vypočteme velikost výsledného obrázku a následně index konkrétního obrázku. Poté vybereme předdefinovaný obrázek a nastavíme mu požadované parametry. Tento obrázek následně vykreslíme. Jednotlivé výsledné přímky pak můžeme upravovat a přepočítáním jejich souřadnic překreslovat.

Element přímky vždy začíná prefixem *line*, podle kterého je možné tyto přímky identifikovat. Všechny přímky jsou umístěny do elementu s identifikátorem spoje, jenž je umístěn jako vrstva pod element mapa, aby tak zařízení překreslovala spodnější vrstvu spojů.

Níže vypsaná funkce nám na základě výpočtu vybere nejvhodnější předem načtený průhledný obrázek přímky v určitém natočení, který dále roztáhne na požadovanou délku. Za upozornění stojí ne zcela často používaný ternární operátor pro první výpočet indexu

obrázku (lineIndex). To nám rozděluje přímky do dvou kategorií, a to v intervalu 0-90°a 91-179°. Zde si musíme uvědomit, že například úhel 45° mezi prvky A a B je zároveň úhlem 225° mezi prvky B a A. Převod jednoduše získáme z minimálních a maximálních souřadnic v jednotlivých složkách. Lze tak říci, že v rozmezí 0 - 90° vykreslujeme přímku z A do B, a v rozsahu 180 - 269° vykreslujeme přímku z B do A a můžeme tak využít stejného obrázku. Alternativně je to i pro úhly 91 - 179°a 271 - 259°.

Poslední zajímavou funkcí je *updateLine* a v ní konstrukce cyklu while. Ta využívá operátor pro bitové operace. Uvedená konstrukce *tmp>>=1* posouvá číslo *tmp* v binární reprezentaci o 1 bit doleva. Vzhledem k faktu, že maximální velikost čísla v proměnné *tmp* může být 256 a velikost obrázků je n-tá mocnina dvojky s maximálním indexem 8 – tedy číslo 256. Číslo v binární reprezentaci tak spadá do jedné z těchto tříd. Výběr natočení původních přímek na 45° a 225° je jednoduše rozlišen na sudých a lichých pozicích pole, ve kterém jsou obrázky předuloženy.

```
function updateLine( lineObjectHandle, Ax, Ay, Bx, By )
{
    var
    xMin = Math.min( Ax, Bx ),
    yMin = Math.min( Ay, By ),
    xMax = Math.max( Ax, Bx ),
    yMax = Math.max( Ay, By ),
    boxWidth = Math.max( xMax-xMin, 1 ),
    boxHeight = Math.max( yMax-yMin, 1 ),
    tmp = Math.min( boxWidth, boxHeight, 256 ),
    lineIndex = (Bx-Ax)*(By-Ay)<0?0:1
    while( tmp>>=1 )
    lineIndex+=2
    lineObjectHandle.src = preloadedImages[lineIndex].src
    with( lineObjectHandle.style )
    {
        width = boxWidth +"px"
        height = boxHeight +"px"
        left = xMin +"px"
        top = yMin +"px"
    }
}
```

9.2. Manipulace s prvky mapy

Další důležitou součástí aplikace je manipulace s jednotlivými zařízeními v mapě. To v aplikaci provádíme přiřazením elementu do třídy drag. Tyto elementy následně mohou být posunovány při držení levého tlačítka myši na elementu a následném přetažení po obrazovce při takzvané metodě drag and drop („táhni a pusť“). Nejprve je třeba potřebným událostem stránky přiřadit patřičné funkce.

```
document.onmousedown = OnMouseDown;
document.onmouseup = OnMouseUp;
document.onclick = grabID;
```

Jedná se o události při stisknutí tlačítka, kliknutí tlačítka a puštění tlačítka. Nejdůležitější funkce jsou popsány v následujících ukázkách. Pro akci zmáčknutí (držení) tlačítka myši předáme událost funkci *OnMouseDown*. Ta v případě, že se jedná o element z třídy *drag,* předá událost pohybu myši funkci *OnMouseMove* a držený element do proměnné *_dragElement*, se kterým pak funkce *OnMouseMove* pracuje tak, že přiřazuje aktuální souřadnice pozice ukazatele myši drženému elementu. Tím vzniká „iluze“ pohybu.

```
function OnMouseMove(e)
{
    if (e == null)
        var e = window.event;
    _dragElement.style.left = (_offsetX + e.clientX - _startX) + 'px';
    _dragElement.style.top = (_offsetY + e.clientY - _startY) + 'px';
    creLink();
}
```

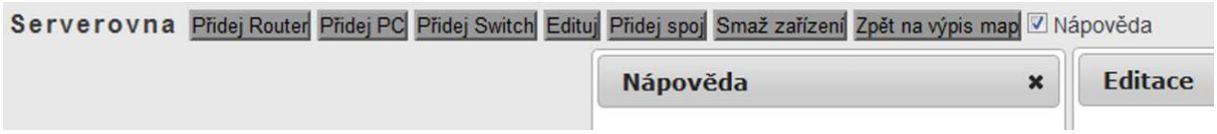
Událost puštění tlačítka myši přiřadíme funkci *OnMouseUp*. Touto funkcí uvolníme událost onmousemove dokumentu, předáme výsledné souřadnice (pomocí knihovny jQuery a technologie AJAX skriptu) serveru. Po jejich uložení uvolníme „držený“ element.

```
function OnMouseUp(e)
{
    if (_dragElement != null)
    {
        _dragElement.style.zIndex = _oldZIndex;
        document.onmousemove = null;
        document.onselectstart = null;
        _dragElement.ondragstart = null;
        jQuery.get("set.php?pos=true", { id: _dragElement.id, x:
        _dragElement.style.left.replace("px",""), y:
        _dragElement.style.top.replace("px","")} );
        _dragElement = null;
    }
}
```

Poslední často používanou událostí je událost *onClick*. Tato událost vyvolá akci při stisku tlačítka a je předána funkci *grabID*. Ta ověří, jestli je v menu označena požadovaná akce (například editace zařízení) a v případě že ano, předá identifikátor požadovaného prvku další funkci. Aby nebyl předáván nahodilý identifikátor kteréhokoli prvku aplikace, provádí se ještě kontrola, zda identifikátor začíná řetězcem ID. Tímto identifikátorem v aplikaci začínají pouze elementy jednotlivých zařízení.

9.3. Menu

Vzhledem k tomu, že v aplikaci je potřeba ovládat řadu funkcí, bylo potřeba pro ovládání mapy implementovat menu. V aplikaci bylo důležité ovládat přidávání jednotlivých zařízení, mazání zařízení, přidávání spojů, editace zařízení. Po implementaci IP dále nastavení IP adres, směrovacích tabulek, jednotlivé vizualizace a další. Původní menu bylo řešeno tlačítky – elementy <button>, jak je vidět na obrázku 9.1.

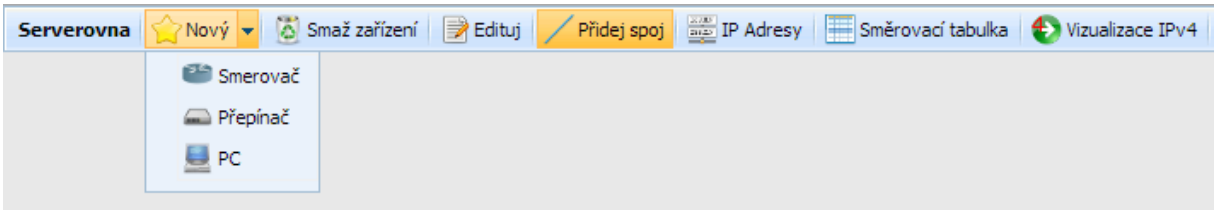


Obrázek 9.1: Původní menu

Při stisknutí tlačítka byla nastavena specifická proměnná na hodnotu *true*, dále pak při stisknutí tlačítka myši na element kontrolována jednotlivá událost a po skončení nastavování proměnná opět nastavena na *false*. Toto řešení bylo z hlediska dalšího rozvoje aplikace velice nevhodné. V případě, že chtěl uživatel například editovat více zařízení za sebou, bylo vždy nutné znovu stisknout požadované tlačítko menu. Dále nebylo graficky znázorněno, které tlačítko bylo stisknuto. Toto by se dalo předělat „přestylováním“ grafiky a přepsáním požadovaných funkcí, avšak z hlediska velikosti celého menu bylo lepším řešením celé menu přepracovat.

Menu bylo možné přepsat a vytvořit pomocí odpovídajících funkcí HTML, CSS a JavaScriptu nebo za pomoci těchto technologií použít některý ze zavedených frameworků pro tvorbu menu. Zvažovaná možnost byla například použít framework Helix [6]', UIZE [14] nebo přímo vytvořit menu pomocí již použitého frameworku jQuery. Avšak zavedení dalších komplexních frameworků by aplikaci dále zkomplikovalo a na pomalejších zařízeních zpomalilo. Využití menu za použití jQuery by bylo možné, avšak úspora především v oblasti místa pro vykreslování by byla minimální.

Z těchto důvodů byla zvolena další zvažovaná varianta a to využití specializované JavaScriptové knihovny, která není tak komplexní jako celý framework. Konkrétně byla použita knihovna DHXMLX toolbar [2] určená výhradně k tvorbě pokročilých menu. Tato knihovna nabízí možnost tvorby menu při využití jednoduchých tlačítek, popisů, přepínačů, přepínacích tlačítek a dalších možností. Z hlediska funkčnosti tato varianta plně vyhovovala požadovaným vlastnostem.



Obrázek 9.2: Ukázka části nového menu

V rámci této JavaScriptové knihovny existuje rozsáhle API, které je i poměrně zdařile zdokumentované. Části dokumentace jsou ovšem obecnějšího charakteru, proto při konkrétní specifické implementaci této knihovny do aplikace bylo zapotřebí často postupovat metodou typu „pokus omyl“.

V menu byla použita rozbalovací tlačítka – na obrázku 9.2 pro nový prvek. Dále oddělovače mezi jednotlivými položkami, které svislou linkou oddělují hranice mezi tlačítky. Nechybí ani textový popisek pro zobrazení názvu mapy a v neposlední řadě přepínací tlačítka. Ta mají definované dva stavy – stisknuto a uvolněno. Tímto mechanismem tak lze nahradit proměnné, které tyto stavy v původním menu zastávaly. Při stisku kteréhokoliv prvku menu jsou navíc generovány události, pomocí nichž lze jednoduše ošetřit například možnost aktuálního stavu stisknutí pouze jednoho tlačítka. Změnou bylo také grafické uspořádání menu včetně zobrazení ikon, v němž se uživatel snáze zorientuje.

9.4. Zpracování dat a serverová část

Jak již bylo zmíněno v kapitole 6 zabývající se návrhem aplikace, zpracování dat probíhá částečně na straně klienta (webový prohlížeč) a částečně na straně serveru. Některá data tak jsou pro potřeby aplikace předzpracována, aby se s výpočtem nemusel zabývat klient. To nastává většinou v případech, kdy by se muselo pro potřeby výpočtu přenést více dat a výsledkem funkcí je pak například pravdivostní výrok.

9.4.1. API

V aplikaci je API využité pro vnitřní komunikaci mezi klientem a serverem. Jedná se o sbírku funkcí vykonávaných na straně serveru zpřístupněných přes jednoduché webové rozhraní. Moderní aplikace dnes pracují s propracovanými API, jako je například REST API. Jedná se o velice propracované a komplexní API, které využívá http metod GET, POST, PUT

a DELETE. Z hlediska složitosti aplikace však bylo použito jednoduché rozhraní, které využívá pouze metodu GET. Jedná se o velice primitivní web API.

Pro zjednodušení a zpřehlednění je API rozděleno do čtyř samostatných skriptů podle účelu volaných funkcí. Jedná se o soubor del.php, který na základě dalších parametrů maže data v databázi. Dále se jedná o soubor add.php pro přidávání příslušných záznamů do databáze. Pro update záznamů v databázi byl vytvořen skript put.php a pro výpis nejrůznějších dat pak load.php. Samotné skripty nejsou jen prostým rozhraním mezi databázovým serverem zpřístupněným přes webové rozhraní, ale obstarávají také zabezpečení příslušných dat.

Struktura API je dále definována konkrétním očekávaným výstupem. První povinný identifikátor u každého požadavku je specifikace dotazu, který musí obsahovat hodnotu *true*. Dále zpravidla následuje identifikátor konkrétního záznamu zasílaný obvykle jako metaproměnná *id*. URI pro získání názvu zařízení při editaci s identifikátorem 33 by tak například vypadlo následovně:

```
load.php?edit_dname=true&id=33
```

Vzhledem k tomu, že JavaScriptový klient v některých případech vrací identifikátor prvku s řetězcem ID, je proto ještě ošetřeno odstranění toho řetězce. Ekvivalentní výsledek by se proto získal i pomocí URI

```
load.php?edit_dname=true&id=ID33
```

Jako odpověď je pak využíván zápis ve tvaru JSON. Například:

```
{ "name": ["Switch Test2","70"]}
```

Formát JSON byl vybrán z důvodu velmi jednoduchého a kvalitního zpracování jak na straně serveru, tak na straně JavaScriptu – respektive s využitím funkcí frameworku jQuery. Jelikož je na straně serveru jako skriptovací programovací jazyk využito PHP, je možné odpověď v JSON formátu generovat dvěma způsoby. První – primitivní - způsob generování JSON formátu je vytvoření příslušného řetězce. Toho lze dosáhnout pomocí funkcí pro výstup – tedy například *echo* nebo *print*. Tato metoda však vyžaduje precizní kontrolu výstupu, aby byla zaručena správná syntaxe formátu JSON. V opačném případě může aplikace vyvolat neočekávané chyby. V aplikaci je této metody taktéž využito při návratu složitějších hodnot z důvodu ladění aplikace.

Dalším způsobem pro získání dat ve formátu JSON v jazyce PHP jsou přímo pro tento formát určené funkce. Jedná se konkrétně o funkce *json_encode* a *json_decode*. Již z názvu funkcí je zřejmé, že první funkce vrací v poli zadaná data ve formátu JSON, kdežto druhá

zmíněná funkce zpracovává jako vstup data ve formátu JSON a vrací pole prvků, kde indexem pole je příslušný název identifikátoru. Vzhledem k tomu, že je formát JSON v aplikaci používán pouze pro přenos dat ve směru server-klient, je aplikací využívaná pouze funkce *json_encode* a jedná se většinou o jednodušší návratové hodnoty – například návratová hodnota true.

```
echo json_encode(array("connected"=>"true"));
```

Nutno však dodat, že funkce pro práci s formátem JSON obsahuje PHP až od verze 5.2, ale vzhledem ke stáří této verze (již není aktivně podporována) a poměrně velké oblibě formátu jsou již tyto funkce běžně dostupné.

Další důležitou funkcí vytvořeného API je zabezpečení dat. Nejedná se o zabezpečení přenášených dat, ale o autorizaci při přístupu k datům a samozřejmě ochrany dat proti nežádoucím zásahům. Jelikož jsou data uložena v DBMS MySQL, při přístupu k nim bylo zapotřebí všechny uživatelsky dostupné vstupy zabezpečit proti zneužití pomocí metody SQL injection. Jedná se o techniku napadení databáze pomocí vložení vlastního SQL dotazu přes neošetřený vstup. V těchto případech záleží na nastavení práv uživatele databáze. V případě vysokého oprávnění uživatele lze dokonce napadnout a poškodit data celého databázového serveru.

Neošetřený MySQL dotaz v php by mohl vypadat například následovně:

```
mysql_query("SELECT * FROM uzivatel WHERE login='".$_GET['user']."';");
```

V takovémto případě lze serveru pomocí metody GET podvrhnout údaje a namísto uživatelského jména zadat například *a' OR 'b'='b* a po doplnění do dotazu bude tento vypadat následovně:

```
("SELECT * FROM uzivatel WHERE login='a' OR 'b'='b '")
```

Takováto podmínka je splněna vždy, a proto se útočník dostane i k datům, ke kterým by neměl.

Pokud bude útočník znát strukturu databáze, může dosazením databázi například smazat nebo jinak poškodit. V případě nepoškození databáze ji může útočník neoprávněně využívat, hlásit se jako jiný uživatel a takovéto uživatele například poškodit.

V PHP lze proti SQL injection využít již připravenou standardní funkci *mysql_real_escape_string()*, která odstraní (nahradí) speciální znaky jazyka SQL z požadovaného řetězce. Použitím této funkce tak například výše zmíněný příklad bude modifikován na řetězec `aâ€  OR â€ bâ€ =â€ b`, který již podmínku nesplní a útočník se tak k datům nedostane.

Jelikož protokol http postrádá kontext o jednotlivých klientech, session v PHP ukládají data o jednotlivých relacích. Po přihlášení se tak údaje o přihlášení uživatele uloží do *session* a následně je pomocí session ověřováno, zda je uživatel přihlášen, a o kterého konkrétního uživatele se jedná. Pomocí session však nelze přímo řídit přístup jednotlivých uživatelů ke konkrétním záznamům databáze.

Z hlediska této aplikace nejsou ukládána kritická data. Jejich pouhé čtení tak není rozhodující. Nejdůležitější proto bylo upravit konkrétní funkce pro manipulaci s daty tak, aby již přihlášený uživatel nemohl přidávat, upravovat nebo mazat data jinému uživateli. Proto při přístupu k těmto funkcím API musí být nejprve ověřeno, zda je uživatel vlastníkem dat, která chce upravovat. V případě, že by takto postupováno nebylo, mohl by si uživatel zjistit i z webového prohlížeče konkrétní tvar URI odesílaného na server a záměrně pozměnit proměnné.

Z hlediska databáze je u každého prvku nutné ověřit, zda je právě přihlášený uživatel jejich vlastníkem. V session je uložen login právě přihlášeného uživatele. Z hlediska databáze je uživatel přímo spjat pouze s mapou. Proto se v případě například mazání IP adresy musí ověřit, jestli vlastníkem mapy, na které je zařízení, které má rozhraní, k němuž je přiřazena IP adresa kterou chceme smazat je opravdu přihlášeným uživatelem, který se o tuto akci pokouší. Až po tomto ověření je možné požadovaná data jakkoliv modifikovat.

Tělo pro mazání rozhraní tak může vypadat následovně:

```
$id = str_replace('ID', '', mysql_real_escape_string($_GET['id']));
$result = mysql_query($sql);
if(mysql_num_rows($result)>0)
{
    mysql_query("DELETE FROM zarizeni WHERE zarizeni.id=".$id);
}
```

9.4.2. Obslužné funkce

Serverová část částečně zpracovává nebo předzpracovává data aplikace. Zpracovánídat se provádí u vstupů, které je zapotřebí před uložením do databáze transformovat. Jedná se tedy například o ukládání IP adres v binárním tvaru, ačkoliv od klienta jsou zasílána ve standardní podobě zápisu IP adres.

Skriptovací jazyk PHP neobsahuje standardní funkce pro kontrolu a převody IP adres. Bylo proto zapotřebí tyto funkce naprogramovat. Vzhledem k odlišnostem zápisu IPv4 a IPv6 adres bylo nutné tyto funkce naprogramovat separátně.

Pro převod standardního zápisu IPv4 adres do binární podoby bylo zapotřebí rozdělit IPv4 adresu na 4 dekadická čísla, tato převést do binárního tvaru již za pomocí standardních PHP funkcí a následně doplnit zleva nulami do délky jednoho bajtu (je-li potřeba). Následně stačí spojit jednotlivé části do výsledného 32bitového řetězce. Obrácený převod pro IPv4 byl jednodušší, kdy se po 8 bitech převádí binární zápis na dekadický a odděluje se tečkami. Stejného principu bylo následně použito pro obdobné převody IPv6 adres. Zde je však nutné podotknout, že stejná IPv6 adresa může být zapsána pomocí více způsobů. Převod ze standardního zápisu do binární formy tak musel být doplněn o transformace zkrácených notací na plnou notaci IPv6 adres a až následné využití obdobného principu jako u IPv4 adres.

Předzpracování se provádí především u získávaných dat, kde je vhodnější zpracovat a přenést vypočtenou hodnotu, nežli data k výpočtu potřebná. Jedná se tak například o hledání a zpracování sousedů jednotlivých prvků nebo jejich dostupnost po IP vrstvě. Při návrhu a programování těchto funkcí bylo nutné vystihnout všechny reálně možné situace a přiblížit jim vracené výsledky funkcí.

Hledání sousedních prvků se provádí pomocí rekurentní funkce *najdi_souseda()*, která vrací pole jednotlivých sousedů. Jedná se o funkci částečně procházející jednoduchý hranově neohodnocený, neorientovaný graf. Částečné proto, že sousedy hledáme pouze na úrovni druhé síťové vrstvy modelu ISO/OSI. Vyhledáváme tedy sousedy připojené na konkrétní rozhraní a v případě, že je na stejné rozhraní daného souseda připojeno další zařízení, rekurentně funkci opakujeme, dokud takovíto sousedé jsou. Funkce jako parametry požaduje identifikátor zkoumaného rozhraní, již projité prvky a rozhraní, ze kterého se ke stávajícímu přišlo. V prvotní inicializaci funkce jsou poslední dva jmenované parametry prázdné řetězce a nabývají hodnoty až při rekurzi.

Pokud by nám však v síti vznikla smyčka, takováto funkce (stejně jako paket v reálné síti) by se zacyklika a „točila“ by se ve smyčce. Funkce proto musí být ukončena v případě, že další procházené rozhraní se nachází v rozhraních již projitých. Všechny navazující funkce proto musí s touto variantou taktéž pracovat.

S funkcí hledání sousedů přímo souvisí funkce *zpracuj_sousedy()*, která k jednotlivým sousedům daného rozhraní přiřazuje jejich IP adresy. Na tuto funkci dále navazuje funkce pojmenovaná *arp()*, v IPv4 a ekvivalentní funkce *nd()*, v IPv6. Tyto funkce načtou IP adresu zkoumaného rozhraní, všechny jeho adresy a následně porovnávají, zda jsou IP adresy rozhraní sousedních zařízení přímo dostupné v rámci podsítí IP adres zkoumaného rozhraní. V případě,

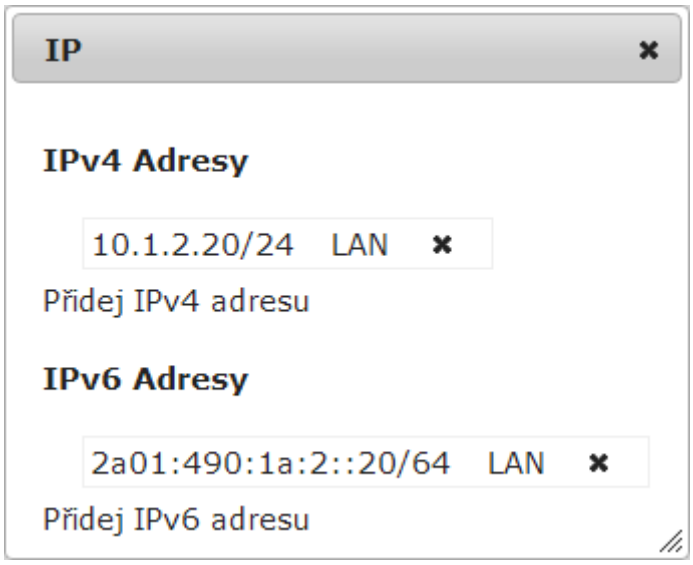
že IP adresa na propojeném rozhraní souseda nepatří do podsítě, ve které je zkoumané rozhraní, není takováto adresa do výsledku funkce zařazena. Funkce následně vrací pole přímo dostupných IP adres.

Tyto funkce jsou využívány při zpracování dat a dotazů na dostupnost adres sousedů a vytváření jednotlivých cest pro animace paketů mezi jednotlivými zařízeními.

9.5. Uživatelská část

Uživatelská část je zaměřena především na graficky přívětivou možnost vytváření a editací jednotlivých uživatelských map a topologií. Při úpravě dat je důležité, aby uživatel nemusel přepínat jednotlivá okna nebo záložky prohlížeče a přesto měl pohodlí nezávislého „okna“, které není statické, ale je možné ho přesouvat v rámci pracovní plochy prohlížeče.

Tuto funkcionalitu zajišťují takzvaná modální dialogová okna, která jsou generována pomocí frameworku jQuery a jQuery UI. Dialogové okno je v tomto případě samostatná oblast okna prohlížeče, kam lze vkládat jakékoliv HTML elementy. V samotné aplikaci pak slouží při nastavování všech důležitých parametrů, jak je patrné z obrázku 9.3, kde je například ukázána možnost nastavování IP adres.



Obrázek 9.3: Výpis IP adres v dialogovém okně

Dialogová okna ve frameworku jQuery a jQuery UI ovládají pomocí metody *dialog()* na specifikovaném HTML prvku. Prvek, který chceme „otevřít“ jako dialogové okno, specifikujeme pomocí atributu *id*. K dialogovému oknu lze definovat celou řadu parametrů. Například zda se jedná o modální dialogové okno, určit požadovanou výšku a šířku okna, jeho

umístění nebo třeba vlastnost, zdali se může přesouvat. Kromě vlastností generuje dialogové okno řadu událostí, které je možné odchytávat. Lze tak případně definovat akci při zavření dialogového okna, při jeho otevření nebo například při jeho posunu. Tyto akce jsou v aplikaci taktéž využívány. Jedná se třeba o přepsání IP adresy v popisku prvku v případě, že je zavřeno dialogové okno pro nastavení IP adresy. Vytvoření jednoduchého dialogového okna v aplikaci vypadá například následovně:

```
jQuery("#dialog").html("");
jQuery("#dialog").attr("title", lang_add_ifc);
jQuery("#dialog").dialog({
       modal: true,
       height: "auto",
       width: "auto"
      });
```

Data nejen do těchto oken jsou nahrávána pomocí rozhraní API. V případě, že se jedná například o IP adresu, jsou tato data přenášena v binární formě. Takto přenesená data je potřeba zobrazit ve standardním zápisu. Proto obdobně jako u serverové části, jsou zde naprogramovány funkce pro převody binární podoby IP adres na běžný zápis. U IPv6 adres je ještě speciální funkce pro zobrazení zápisu v kompletní nebo zkrácené notaci.

Stejně jako je nutné data zobrazovat, je i nutné tato data ukládat. Pro prvotní kontrolu při ukládání dat je zapotřebí zkontrolovat a případně uživatele upozornit, že zadaná data nejsou v daném kontextu platná. V případě IP adres tak například zjistit, zdali je tato adresa zadána ve správném formátu.

Jak již bylo zmíněno, data jsou se serverem vyměňována pomocí jednoduchého webového API. K přístupu k tomuto API se využívá metoda AJAX. Pravdou ovšem je, že data nemusí být přenášena asynchronně a jako formát přenosu nemusí být využíváno formátu XML. Aplikace pro přenos dat využívá formátu JSON a obou dvou přístupů k nim – tedy jak synchronního tak asynchronního. Vhodnost obou přístupů pak záleží na konkrétní implementované funkci.

Celý proces práce pomocí technologie AJAX je obstaráván sadou funkcí a metod frameworku jQuery. Asynchronní přístup je využíván například při prostém výpisu požadovaných dat. Není zde potřeba čekat na výsledek funkce. Pro asyncrhonní přístup se v aplikaci používá metoda *getJSON()*, která následně přijatá data zpracovává jako pole prvků.

```
jQuery.getJSON("load.php?ipv6route=true&id="+theId, function(data) {
jQuery.each(data, function(key, val) {
jQuery("#ipv6route").append(val[0] + "\" + val[1]);
        });
      });
```

Tento přístup se ovšem stává nevhodným ve chvíli, kdy například potřebujeme získaná data využívat jako návratovou hodnotu funkce. Při asynchronním přístupu by takováto funkce nečekala na vrácená data, ale skončila by dříve a nevrátila by požadovaná data. V takovémto případě je tedy synchronní přístup nutný. Příkladem takové funkce může být například získání názvu rozhraní na základě jeho identifikátoru. Asynchnronnímu přístupu zde zabráníme pomocí parametru async, kterému nastavíme příznak *false*, jak je vidět v následujícím příkladu.

```
function ifname(id)
{
     var scriptUrl = "load.php?ifname=true&id="+ id;
     jQuery.ajax({
        url: scriptUrl,
        type: 'get',
        dataType: "json",d
        async: false,
        success: function(data) {
            result = data.nasdme;
        }
     });
     return result;
}
```

Jelikož jsou data načítána dynamicky, je také potřeba tato data v okně prohlížeče zobrazit bez nutnosti tuto zobrazovanou stránku znovu načíst celou. Naopak například při mazání prvků je nutné tyto prvky smazat opětovně bez nutnosti znovunačtení stránky. K tomu se v aplikaci opětovně využívá metod frameworku jQuery.

Pro přidávání slouží metoda *append()* a pro odebírání obsahu metoda *empty()*. Metoda append přidává požadovaný HTML kód na konec obsahu specifikovaného elementu. Element specifikujeme pomocí určení jeho identifikátoru, tedy parametru *id* HTML prvku. Tento identifikátor musí být v celé aplikaci jedinečný. V opačném případě nefunguje funkce append korektně. Metoda empty funguje obdobným principem, avšak obsah vnořený do požadovaného elementu smaže.

```
jQuery("#ID").empty().append("<div>Text</div>");
```

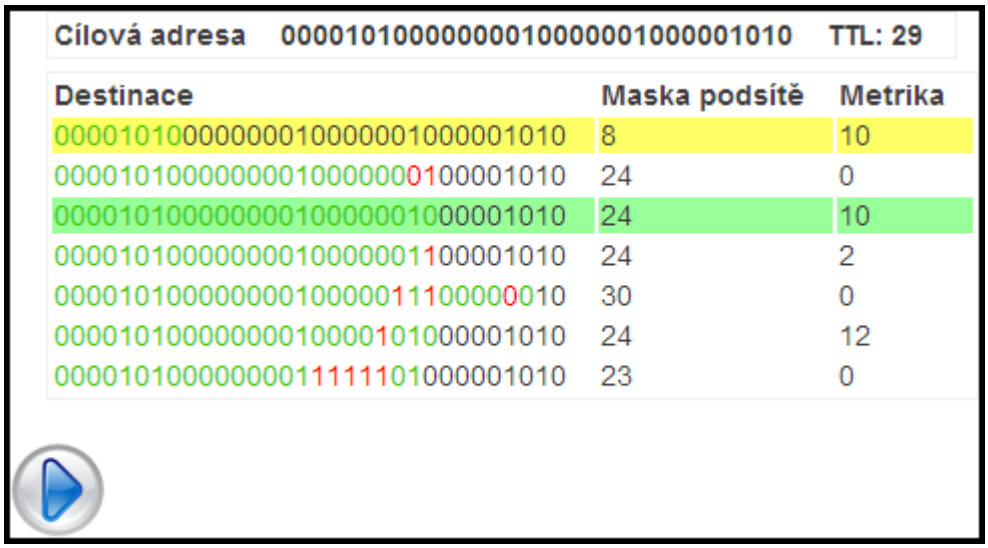
Tato ukázka nejprve vymaže obsah elementu *ID* a následně vypíše jako vnořený element HTML kód * Text *. Pomocí těchto metod je generován veškerý dynamický obsah aplikace, všechna dialogová okna, zařízení, spoje a další.

9.6. Simulace

Pro simulaci bylo zvažováno několik verzí zobrazení. Jednalo se především o návrh vhodného způsobu zobrazení výběru ze směrovacích tabulek. První zamýšlená verze byla naprogramována jako souvislá animace.

Celková animace začínala ve zdrojovém prvku, kde byla spuštěna animace výběru ze směrovací tabulky. Zde byla zobrazena cílová adresa a všechny řádky směrovací tabulky. Směrovací tabulku bylo možné zobrazit jak v podobě standardního zápisu, tak v podobě binární. Element cílové adresy se následně pohyboval po jednotlivých řádcích. V případě shody bylo písmo na tomto řádku označeno zeleně v opačném případě červeně. Pokud byl zároveň řádek vyhodnocen jako nejlepší shoda, byl podklad řádku podbarven žlutě. Po skončení animace byl výsledný záznam ze směrovací tabulky zvýrazněn, směrovací tabulka byla zavřena a vizualizace pokračovala animací paketu mezi příslušnými prvky nebo chybovou hláškou.

Tento způsob vizualizace se však neosvědčil, jelikož zde nebylo možné jednotlivé kroky pozastavit. Pro začínající studenty by tak toto řešení nebylo vhodné. Značně nepřehledné až zavádějící by taktéž mohlo být relativně rychlé procházení směrovacích tabulek. Nastavení optimální rychlosti by bylo značně závislé na konkrétním studentovi.



Obrázek 9.4: Zobrazení výběru ze směrovací tabulky v binární podobě

Výsledné řešení proto bylo zvoleno tak, aby mezi jednotlivými kroky byla celá animace pozastavena. Výběr ze směrovací tabulky byl zjednodušen pouze na prosté zobrazení tabulky, kdy zeleně je podbarven řádek, který vyhovuje použité nejlepší cestě a žlutě jsou podbarveny řádky, které sice cílové adrese vyhovují, ale nejedná se zde o nejlepší shodu (jak je patrné z obrázku 9.4). V případě přepnutí do binární podoby je navíc zobrazeno, ve kterých bitech se

daná síť shoduje a ve kterých se naopak liší. Student nebo vyučující má tak k dispozici libovolné množství času pro studium nebo vysvětlení jednotlivých kroků.

V aplikaci bylo nutné, aby jednotlivé kroky animace navazovaly a probíhaly v jisté po sobě jdoucí sérii. Vzhledem k faktu, že k simulaci bylo použito frameworku jQuery, kde jsou jednotlivé animace tvořeny jako asynchronní funkce, bylo nutné pro účely animací tyto funkce serializovat.

Všechny metody sloužící v jQuery pro animaci prvků obsahují taktéž takzvané callback funkce (funkce zpětného volání). Hlavní funkci je zde možné jako parametr předat jakoukoli další funkci, která bude provedena po úspěšném dokončení funkce hlavní. Tímto způsobem lze funkce zřetězit takovým způsobem, kdy funkce A má jako callback funkci B, která má callback funkci A. Další možností je použití přímo rekurentního volání jakožto callback funkce. V obou případech je však nutné nastavit ukončující podmínky v rámci některé z funkcí tak, aby nedošlo k nekonečnému cyklu. Základní kroky při použití těchto funkcí jsou obdobné jako u rekurzivních funkcí.

```
function animace_polem(pole)
{
  ...
  for(var i=2; i<prvky.length; i++)
  {
      zkracene_pole = zkracene_pole+","+prvky[i];
  }
  var imgTag = document.createElement("img");
  ...
  var draha = Math.ceil(Math.sqrt(hor*hor+ver*ver));
  var cas = Math.ceil(draha*17);
    jQuery("#packet").animate({
    left: position2.left+"px",
    top:  position2.top+"px"
  }, cas , function(){
      if(prvky.length == 2)
        {
          jQuery('#packet').fadeOut(800, function(){
              jQuery('#packet').remove();
          });
        }
        else
        {
            jQuery('#packet').remove();
            animace_polem(zkracene_pole);
        }
      }
    );}
```

Vhodnou ukázkou je funkce pro animaci průchodu paketu přes několik prvků s názvem *animace_polem()*. Jako parametr je předáváno pole prvků, kterými daný paket prochází. Vždy se animuje mezi prvními dvěma prvky pole. První prvek je z tohoto pole pro další krok rekurze

odebrán. Tím splňujeme jeden ze základních předpokladů rekurze, kdy bychom měli daný problém systematicky zjednodušovat. Rekurzivní volání končí ve chvíli, kdy animujeme mezi dvěma posledními prvky tohoto pole. V této funkci ještě stojí za povšimnutí proměnná *cas*, která slouží k výpočtu času, po který se má daný paket přesouvat mezi dvěma body. Ve výsledku je tak rychlost posunu paketu stejná pro libovolnou vzdálenost.

Výsledná celková animace probíhá od samotného spuštění v několika navazujících krocích. Na základě algoritmu je nejprve vyhodnoceno, zda se cílová adresa neshoduje s některou adresou stávajícího prvku. Pokud ano, animace zde skončí a paket plynule zmizí. Pokud ne, je zkoumáno, zda IP adresa cíle nepatří do rozsahu přímo připojených sítí. V případě, že ano, je následně zkoumáno, zdali je cíl přímo jako sousedící uzel a zároveň je dostupný. Pokud dostupný není, je uživatel opětovně informován o nedostupnosti cílového hostitele. Pokud cílová adresa nespadá do rozsahů přímo připojených, vyhledává se ve směrovací tabulce, která je zároveň zobrazena. Dále je potřeba, aby uživatel spustil animaci stisknutím tlačítka „PLAY“. V případě, že není žádný záznam vhodný, je animace ukončena a uživatel je informován o nedostupnosti cílové sítě. V opačném případě pokračuje animace a na dalším prvku se celý výběr opakuje.

9.7. Export a import map

Jelikož by měla být aplikace primárně určena pro studenty základů počítačových sítí, bylo vhodné v aplikaci umožnit výsledné mapy a jejich topologie nejenom ukládat, ale taktéž exportovat a sdílet. V případě, že vyučující například využije ukázky v aplikaci, je následně možné připravenou mapu poskytnout studentům k dispozici. Studenti poté mohou ukázkové příklady podrobněji studovat.

Teoretickou možností implementace by bylo prosté exportování SQL kódu z databáze MySQL. Při této variantě však vzniká problém s jedinečnými identifikátory. Nehledě na fakt, že tímto bychom rozkryli celou původní strukturu identifikátorů. Pro nezávislý export a import map se tak proto přímo nabízí využití jazyka XML. Zde je možné v jednoduchém textovém formátu přenést celou výslednou mapu.

Z hlediska návrhu databáze je patrné, že největší problém v tomto případě vzniká při přenosu jedinečných identifikátorů jednotlivých prvků a následné využití těchto identifikátorů jako cizích klíčů. Je zde proto zapotřebí zachovat informace o jednotlivých vazbách.

Protože je databáze tvořena jakousi stromovou strukturou, je nutné postupovat od kořenového prvku – v tomto případě tabulky mapa, jak je patrné z obrázku 6.1. Dále se ukládají jednotlivá zařízení. Těm postupně přiřazujeme identifikátory začínající zkratkou anglického názvu elementu, které jsou dále následovány číslicí a ukládáme je do pole, jehož indexem je identifikátor originální. Stejně se postupuje při záměně všech ostatních identifikátorů. V případě, že potřebujeme zaměnit identifikátor jako cizí klíč, stačí vzít položku pole, jejíž index odpovídá právě původní hodnotě. Tímto způsobem lze zachovat kompletní informační hodnotu, aniž bychom zároveň rozkryli původní strukturu identifikátorů.

Jelikož je struktura XML souboru uložena jako prostý text, je možné tento soubor editovat libovolným textovým editorem. Přímo se tak nabízí, aby i exportovaná data byla čitelná pro běžné uživatele, avšak některé informace v databázi jsou ukládány v binární podobě. Je proto zapotřebí tato data převést do standardního zápisu.

V případě importu je obecně postupováno obráceným způsobem. Nejprve je vložena nová mapa, jejíž skutečný identifikátor je uložen jako proměnná. Následně jsou vkládána jednotlivá zařízení, jejichž skutečné identifikátory jsou opětovně ukládány do pole, kde indexem je tentokrát původní identifikátor z XML. Dále jsou prvky vkládány do databáze a v případě cizích klíčů jsou tyto opět nahrazovány z indexovaného pole. V případě jednotlivých adres je ještě nutné zkontrolovat, zda adresy vyhovují zápisu a poté tyto adresy převést do binární podoby. Struktura uložené mapy je definována pomocí DTD dokumentu, který je uveden jako příloha 2 této práce a je taktéž obsažen na přiloženém CD.

10. Testování

Testování výsledné aplikace probíhalo ve dvou fázích. V prvotní fázi při tvorbě aplikace, kdy byla prakticky ověřována funkčnost jednotlivých funkčních celků a následně po dokončení aplikace při testování kompatibility aplikace na různých zařízeních.

10.1. Při tvorbě aplikace

Při tvorbě byla aplikace převážně testována na kompatibilitu ve webových prohlížečích. Jako primární testovací prostředí byl využíván prohlížeč Google Chrome. Protože aplikace obsahuje velké množství dynamických prvků, které jsou generovány až za běhu, bylo využito prostředí *Nástroje pro vývojáře*.

Tento vývojářský nástroj obsahuje všechny nejdůležitější prvky pro ladění webové aplikace. Jsou zde ale také další velice užitečné nástroje, jako například výpis všech událostí, které se váží na vybraný HTML element, jednotlivé načtené skripty nebo časy jednotlivých požadavků. Samozřejmostí je také úprava kódu v rámci prohlížeče bez zásahu do vyvíjené aplikace. U tohoto vývojářského nástroje je možné například změnit parametr požadovaného HTML elementu nebo změnit jeho styl.

Další důležitou vlastností ladicího nástroje je možnost kontroly zdrojů a požadavků. Při aplikaci dochází k výměně dat s webovým serverem pomocí technologie AJAX. V případě ladicího nástroje je možné procházení jednotlivých požadavků. Je zde velice jednoduché kontrolovat přijímané hodnoty pomocí JSON zápisu. Celé ladění navíc doplňuje nástroj na sledování síťové komunikace aplikace. Zde je vidět nejen konkrétní dotaz, ale i výsledný status, velikost a čas zpracování. Je proto možné optimalizovat přenášená data a rychlost jejich zpracování.

Při testování byla pozornost věnována především rychlosti a funkcionalitě aplikace. Funkcionalita aplikace byla testována na základních modelových případech počítačových sítí. Jednalo se tak především o různé varianty plochých sítí, smyček v plochých sítích nebo výsledky směrování a vhodného výběru při směrování. Ve všech uvažovaných možnostech aplikace vyhověla nebo byla následně upravena do funkční podoby, aby se co nejvíce přiblížila chování reálných počítačových sítí.

Pro zvýšení rychlosti a odezvy byla především optimalizována část na straně serveru, aby bylo minimalizováno množství přenášených dat. Z hlediska optimalizace jednotlivých

předávaných hodnot při výměně dat pomocí technologie AJAX byla datová úspora v řádech jednotek až desítek bajtů a pro změření zrychlení odezvy by se toto měření pohybovalo na hranici chyby měření.

Další optimalizací pak byla komprese jednotlivých skriptů. Jednalo se o kompresi JavaScriptových knihoven a kaskádových stylů (CSS). Pro kompresi bylo využito online verze kompresoru YUI [15]. Kompresor odstraní všechny bílé znaky a částečně nahradí proměnné tak, aby bylo dosaženo co nejmenšího objemu přenášených dat. Lokální proměnnou *pole* tak například nahradí za proměnnou *a*, čímž se ušetří 3 znaky. Kompresní poměr se pohyboval od 37% do 46%, čímž se podařilo celkově zmenšit tato data o 49,3Kb.

Po testování byla ještě provedena optimalizace pro nejvýznamnější webové prohlížeče. Optimalizace probíhala pro Internet Explorer a Mozilla Firefox a dále pro integrovaný prohlížeč mobilní platformy Android. V rámci optimalizací pro všechny webové prohlížeče nebylo možné dodržet validity celkového HTML kódu dle standardů W3C. V některých případech bylo například i nutné přímé vkládání stylů do HTML kódu namísto do CSS. Výsledná aplikace tak není validní, ale optimalizovaná pro správnou funkčnost a zobrazení v hlavních webových prohlížečích.

10.2. Testování kompatibilních zařízení

Po dokončení byla výsledná aplikace testována v dalších prohlížečích a na dalších platformách. Ze zajímavějších funkčních platforem to byly například mobilní telefony nebo televizory. Některé drobné nedostatky byly z aplikace odstraněny nebo byla aplikace optimalizována. Aplikace byla úspěšně otestována v aplikacích a platformách, které jsou vypsány v tabulce 10.1.

Platforma	Operační systém	Prohlížeč
x86	Windows XP, 7, 8	Google chrome 23.0-26.0
X86	Windows XP, 7, 8	Mozilla Firefox 17.0-20.0
X86	Windows XP, 7, 8	Internet Explorer 7-10
X86	Windows XP, 7, 8	Opera 12
Mobilní telefon/tablet	Android 2.3, 4.1	Integrovaný
Mobilní telefon/tablet	Android 2.3, 4.1	Opera Mobile
Mobilní telefon/tablet	Android 4.1	Chrome
Mobilní telefon/tablet	Android 4.1	Mozilla Firefox
Mobilní telefon Nokia	Symbian 3	Integrovaný
Televizor	LG LW650S	Integrovaný
Televizor	Samsung	Integorvaný

Tabulka 10.1: Kompatibilní prohlížeče (testováno duben 2013)

11. Závěr

V rámci diplomové práce se povedlo vytvořit webovou aplikace pro vizualizaci průchodu paketu počítačovou sítí včetně znázornění průběhu směrování. V průběhu řešení diplomové práce se podařilo navrhnout a vytvořit webovou aplikaci včetně odpovídajících vizualizací. Při praktickém řešení nastalo několik zásadních problémů, které se však podařilo vyřešit.

Výsledná aplikace je funkční jak v moderních webových prohlížečích, tak ve starších verzích stále oblíbeného prohlížeče Internet Explorer. Výhodou zpracování je také kompatibilita s nestandardními zařízeními, jako jsou například televizory s webovým prohlížečem.

Aplikaci by bylo možné dále vylepšit o podporu dynamického směrování. K tomu by mohly být implementovány dnes velmi hojně využívané protokoly OSPF, RIP a BGP. V rámci vylepšení by bylo možné rozšířit aplikaci o protokoly dalších vrstev. Například na spojové vrstvě by bylo vhodné implementovat protokoly pro odstraňování smyček. Jednalo by se pak především o spanning tree protokol, popřípadě i jeho vylepšující varianty, nebo nové protokoly jako shortest path bridging případně „konkurenční“ TRILL.

Z aplikační vrstvy by pro jednodušší konfiguraci adres bylo vhodné implementovat mechanismus jejich dynamického přidělování, ať již v podobě DHCP pro IPv4, nebo například bezstavovou automatickou konfiguraci pro IPv6. Zajímavé by bylo například zobrazení výměny paketů včetně jejich obsahu při dotazech DNS. Ve výčtu případných vylepšení aplikace by bylo možné dále pokračovat. Aplikace by se tak stala komplexním vizualizačním a simulačním nástrojem pro výuku a experimentování nejenom pro začátečníky, ale i pro síťové administrátory.

Výhodou zvoleného řešení a konkrétní implementace je, že další možnosti rozšíření by bylo možné implementovat bez zásadnějších zásahů do základu aplikace, k čemuž přispívá zpracování IP adres v binární podobě.

Vhodným využitím stávající aplikace mohou být kurzy základů počítačových sítí, kde by aplikace sloužila jako učební pomůcka. Aplikaci by bylo též možné použít jako základní a velmi jednoduchý dokumentační nástroj menších počítačových sítí. Jako pozitivum lze v obou případech zdůraznit možnost ukládání a sdílení vytvořených schémat.

Seznam použité literatury

[1] CISCO SYSTEMS. *Cisco Packet Tracer - Cisco Systems* [online]. [cit. 2013-04-12]. Dostupné z: http://www.cisco.com/web/learning/netacad/course_catalog/PacketTracer.html

[2] DHTMLX LTD. *JavaScript DHT*ML Toolbar with Rich Client-Side API - dhtmlxToolbar [online]. © 1998-2013 [cit. 2013-04-11]. Dostupné z: http://dhtmlx.com/docs/products/dhtmlxToolbar/

[3] DITTRICH, Vít. *Webové grafické rozhraní pro vytváření schémat počítačových sítí*. Liberec, 2012. Magisterský projekt. Technická Univerzita v Liberci. Vedoucí práce doc. RNDr. Pavel Satrapa, Ph.D.

[4] GNS3. *Graphical Network Simulator - GNS3* [online]. © 2007-2013 [cit. 2013-04-23]. Dostupné z: http://www.gns3.net/

[5] MATHIEU, Henri. *Drawing lines in JavaScript | www.p01.org* [online]. 2001 [cit. 2013-04-08]. Dostupné z: http://www.p01.org/releases/Drawing_lines_in_JavaScript/

[6] JOOMSHAPER. Helix Framework [online]. © 2010 - 2011 [cit. 2013-04-10]. Dostupné z: http://helix.joomshaper.com/

[7] MESSERMAN, Gil, Gilad KARNI a Uri BRAUN. *IP Routing Simulation* [online]. [2006] [cit. 2013-04-23]. Dostupné z: http://cisx1.uma.maine.edu/~wbackman/cis341/sims/ip/ip.html

[8] NETMONITOR - SPIR - GEMIUS & MEDIARESEARCH. Měsíční zpráva - Únor 2013. In: *NetMonitor* [online]. 2012 [cit. 2013-04-05]. Dostupné z: http://www.netmonitor.cz/sites/default/files/vvnetmon/2013_02_netmonitor_offline_report.pdf

[9] PETERKA, Jiří. Síťový model TCP/IP. *Jiří Peterka: archiv článků a přednášek* [online]. © 2011 [cit. 2013-04-07]. Dostupné z: http://www.earchiv.cz/a92/a231c110.php3

[10] SATRAPA, Pavel. IPv6: internetový protokol verze 6. 3., aktualiz. a dopl. vyd. Praha: CZ.NIC, c2011, 407 s. CZ.NIC. ISBN 978-80-904248-4-5.

[11] SMOTLACHA, Vladimír. *Linková vrstva, metody přístupu* [online]. Praha, 2011 [cit. 2013-04-23]. Dostupné z: https://edux.fit.cvut.cz/oppa/BI-PSI/prednasky/p3-link.pdf. Přednáška. České vysoké učení technické v Praze.

[12] STATCOUNTER. *StatCounter - Free Invisible Web Tracker, Hit Counter and Web Stats* [online]. © 1999-2013 [cit. 2013-04-05]. Dostupné z: http://www.statcounter.com

[13] THE JQUERY FOUNDATION. *JQuery* [online]. © 2013 [cit. 2013-04-09]. Dostupné z: http:// www.jquery.com

[14] UIZE JAVASCRIPT FRAMEWORK. *UIZE JavaScript Framework | AJAX, RIA, widgets, JSON, OOP, Class Inheritance, XMLHttpRequest, D*OM manipulation, and all that stuff [online]. © 1997-2013 [cit. 2013-04-10]. Dostupné z: http://www.uize.com/

[15] YAHOO!. *Online YUI Compressor* [online]. © 2008-2012 [cit. 2013-05-07]. Dostupné z: http://refresh-sf.com/yui/

Příloha 1 – obsah přiloženého CD

Přiložené CD obsahuje adresáře s daty:

- DP_Vit_Dittrich.pdf dokument diplomové práce ve formátu PDF
- DP_Vit_Dittrich.tar.gz archiv se zdrojovými kódy aplikace

Příloha 2 – definice struktury XML exportu v DTD

```
<!ELEMENT   mapa                    (template*, zarizeni*, spoj*)>
<!ELEMENT   template                (#PCDATA)>
<!ELEMENT   zarizeni                (rozhrani*)>
<!ELEMENT   spoj                    (#PCDATA)>
<!ELEMENT   rozhrani
(ipv4*,ipv6*,routovacitabulkav4*,routovacitabulkav6*)>
<!ELEMENT   ipv4                    (#PCDATA)>
<!ELEMENT   routovacitabulkav4      (#PCDATA)>
<!ELEMENT   ipv6                    (#PCDATA)>
<!ELEMENT   routovacitabulkav6      (#PCDATA)>
<!ATTLIST   mapa
            nazev           CDATA   #REQUIRED>
<!ATTLIST   template
            id              CDATA   #REQUIRED
            typ             (0|1|2) #REQUIRED
            nazevrozhrani   CDATA   #REQUIRED>
<!ATTLIST   zarizeni
            id              CDATA   #REQUIRED
            nazev           CDATA   #REQUIRED
            router          (0|1|2) #REQUIRED
            x               CDATA   #REQUIRED
            y               CDATA   #REQUIRED>
<!ATTLIST   rozhrani
            id              CDATA   #REQUIRED
            nazev           CDATA   #REQUIRED
            zarizeniid      CDATA   #REQUIRED>
<!ATTLIST   ipv4
            id              CDATA   #REQUIRED
            rozhraniid      CDATA   #REQUIRED
            ipv4            CDATA   #REQUIRED
            maska           CDATA   #REQUIRED>
<!ATTLIST   routovacitabulkav4
            id              CDATA   #REQUIRED
            rozhraniid      CDATA   #REQUIRED
            network         CDATA   #REQUIRED
            maska           CDATA   #REQUIRED
            brana           CDATA   #REQUIRED
            metrika         CDATA   #REQUIRED>
<!ATTLIST   ipv6
            id              CDATA   #REQUIRED
            rozhraniid      CDATA   #REQUIRED
            ipv6            CDATA   #REQUIRED
            maska           CDATA   #REQUIRED>
<!ATTLIST   routovacitabulkav6
            id              CDATA   #REQUIRED
            rozhraniid      CDATA   #REQUIRED
            network         CDATA   #REQUIRED
            maska           CDATA   #REQUIRED
            brana           CDATA   #REQUIRED
            metrika         CDATA   #REQUIRED>
<!ATTLIST   spoj
            idparent        CDATA   #REQUIRED
            idchild         CDATA   #REQUIRED>
```