VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV INTELIGENTNÍCH SYSTÉMŮ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF INTELLIGENT SYSTEMS

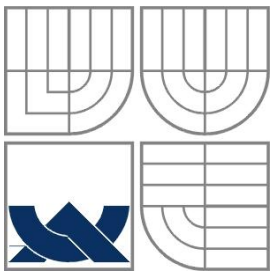
APLIKACE PRO VZDÁLENOU SPRÁVU MIKROTIK ROUTEROS

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

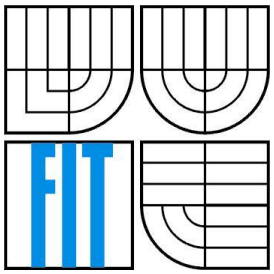
AUTOR PRÁCE
AUTHOR

JIŘÍ DOLEŽAL

BRNO 2012



VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV INTELIGENTNÍCH SYSTÉMŮ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF INTELLIGENT SYSTEMS

APLIKACE PRO VZDÁLENOU SPRÁVU MIKROTIK ROUTEROS

APPLICATION FOR MIKROTIK ROUTEROS REMOTE MANAGEMENT

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE — JIŘÍ DOLEŽAL
AUTHOR

VEDOUCÍ PRÁCE — Ing. JOSEF HÁJEK
SUPERVISOR

BRNO 2012

Abstrakt

Tato bakalářská práce se zabývá návrhem a tvorbou aplikace pro vzdálenou správu MikroTik RouterOS. V rámci práce byla vytvořena aplikace umožňující konfiguraci síťových služeb směrovačů. Práce obsahuje návrh a implementaci aplikace v programovacím jazyce C#. Použitím frameworku Mono bylo dosaženo přenositelnosti aplikace mezi operačními systémy MS Windows a Linux. Aplikace využívá přehledné grafické uživatelské rozhraní pro snadnou správu zařízení s MikroTik RouterOS.

Abstract

This bachelor´s thesis deals with design and implementation of an application for remote management MikroTik RouterOS. Within this thesis has been created an application which allows to configure network router services. The thesis contains design and implementation of an application written in programming language C#. The application is runnable under operation systems MS Windows and Linux by using framework Mono. The application uses a user friendly graphical interface for simple management devices with MikroTik RouterOS.

Klíčová slova

MikroTik, RouterOS, RouterBoard, vzdálená správa, API, grafické uživatelské rozhraní, .NET, C#, Mono.

Keywords

MikroTik, RouterOS, RouterBoard, remote management, API, graphical user interface, .NET, C#, Mono.

Citace

Doležal Jiří: Aplikace pro vzdálenou správu MikroTik RouterOS, bakalářská práce, Brno, FIT VUT v Brně, 2012

Aplikace pro vzdálenou správu MikroTik RouterOS

Prohlášení

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně pod vedením pana Ing. Josefa Hájka. Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

……

Jiří Doležal

7. května 2012

Poděkování

Rád bych poděkoval vedoucímu bakalářské práce panu Ing. Josefu Hájkovi za odbornou pomoc, ochotu a čas, který mi věnoval při tvorbě této práce.

Obsah

1 Úvod

S rozvojem počítačových sítí narůstá i počet směrovačů v síti a tím narůstají i požadavky na správu těchto zařízení. Správu již neprovádí pouze školení správci sítí, ale mnohdy i zdatnější uživatelé PC z prostředí různých operačních systémů. Směrovač již není chápán jen jako prvek, který rozděluje větší sítě na menší segmenty, ale často slouží k připojení malých privátních WIFI sítí se sítí Internet. Proto jej můžeme nalézt téměř v každé domácnosti. Tím vzniká potřeba jednoduchého a intuitivního uživatelského rozhraní pro správu zařízení.

V poslední době zaznamenaly velký rozmach mezi uživateli zařízení firmy MikroTik. Tato zařízení se vyznačují velmi dobrým výkonem a nízkou cenou za poskytované funkce. Směrovače MikroTik ocení nejen běžní uživatelé, ale i malé firmy, které potřebují propojit své pobočky za minimální náklady.

Cílem mé bakalářské práce je navrhnout a vytvořit aplikaci pro správu zařízení se systémem RouterOS, která bude přenositelná mezi operačními systémy Linux a MS Windows. Práce se skládá z osmi kapitol.

Druhá kapitola je teoretická, určená k zopakování základních znalostí o směrovačích a směrování v počítačových sítích.

Třetí kapitola představuje firmu MikroTik a některé z jejich produktů. Z hardwarových produktů jsou zde představeny některé modely RouterBoardů. Ze softwaru je zde popsán RouterOS spolu se způsoby jeho konfigurace.

Čtvrtá kapitola popisuje protokol API (Application Programming Interface) firmy MikroTik a osvětluje základní příkazy pro konfiguraci RouterBoardů.

Pátá kapitola je zaměřena na návrh aplikace pro vzdálenou správu směrovačů MikroTik. V této kapitole je popsáno uživatelské rozhraní aplikace, vymezení implementovaných funkcí a jejich teoretický rozbor.

Šestá kapitola se zabývá vlastní implementací aplikace a použitými technologiemi.

Sedmá kapitola je věnována praktickému ověření funkčnosti aplikace a metodikám testování aplikace.

Poslední kapitolou je závěr, kde je zhodnocena implementovaná aplikace. Dále je zde nástin možného rozšíření funkcionalit aplikace.

2 Směrovač

Tato kapitola popisuje základní fakta o směrovačích a jejich činnosti.
Směrovač (router) je aktivní prvek v počítačových sítích, který je schopen propojit počítačové sítě různých topologií. Pracuje na třetí vrstvě modelu ISO/OSI (viz obrázek 2.1), tedy na vrstvě síťové. Ke směrování využívá informace přenášené v záhlavích paketů. Směrovač má dvě základní úlohy:

- Určit nejvhodnější cestu sítí od zdroje k cíli v závislosti na stanoveném kritériu.
- Směrování paketů ze vstupů na výstupy dle směrovací tabulky.

Obrázek 2.1: Model ISO/OSI.

Směrování je proces, kdy se na základě záznamů obsažených ve směrovací tabulce určuje cesta datagramu počítačovou sítí.
Směrovací tabulka typicky obsahuje informace:

- Číslo cílové podsítě – IP adresa podsítě.
- Maska podsítě – společně s číslem podsítě vymezuje rozsah platných IP adres.
- Výchozí brána – IP adresa směrovače, kterému má být datagram předán, pokud podsíť není přímo dosažitelná.
- Síťové rozhraní – rozhraní směrovače, skrze které se má paket odeslat.

Podle způsobu vzniku záznamů ve směrovací tabulce hovoříme o statickém nebo dynamickém směrování. Směrovací tabulka obsahuje záznamy jen o dostupných sítích (nikoli o všech připojených stanicích) a odpovídajících portech směrovače, které se používají pro směrování paketů. Směrovací tabulka je tedy založena na znalosti logického uspořádání sítě.

Na rozdíl od přepínačů (switch) směrovač implicitně blokuje všechny pakety, u kterých rozezná cílovou adresu typu broadcast. Pokud směrovač obdrží paket, jehož cílovou IP adresu nemá ve své směrovací tabulce, paket je zahozen nebo odeslán na tzv. implicitní síť.

Další z funkcí směrovače je kontrola a změna doby životnosti (TTL) příchozích paketů. Hodnota TTL[1] udává počet směrovačů, kterými může být daný paket zpracován. Zpracováním paketu směrovačem se sníží hodnota TTL o jedničku. Pokud je hodnota TTL obdrženého paketu nulová, bude paket směrován pouze v případě, že se směrovač nachází v síti, ve které je zařízení s cílovou IP adresou. V opačném případě směrovač paket zahodí a vygeneruje chybovou zprávu.

[1] *Time To Live* – doba platnosti dat nebo počet průchodů aktivními prvky.

3 Produkty firmy MikroTik

Tato kapitola uvádí informace o firmě MikroTik a jejich produktech. Při sestavování těchto informací jsem čerpal z oficiálních webových stránek této společnosti [5].

3.1 Společnost MikroTik

Společnost byla založena roku 1995 v Lotyšsku. V roce 1997 tato společnost vydala operační systém pod názvem RouterOS, který umožňoval správu různých směrovacích zařízení. Od roku 2002 firma MikroTik vyrábí i vlastní hardware, který se nazývá RouterBOARD [15]. Jedná se o směrovací zařízení, která se používají v LAN i WIFI sítích. RouterBOARDy se staly oblíbenými především díky jejich výkonu, stabilitě a v neposlední řadě i nízké ceně.

3.2 RouterBOARD

V současnosti je na trhu několik desítek modelů RouterBOARDů, které se liší především výkonem, vybavením, licencí RouterOS a možnostmi rozšíření.

Jako příklad uvádím tyto modely: RB 411, RB 433, RB 711, RB 750 nebo RB 1100. Parametry uvedených modelů jsou uvedeny v tabulce 3.1. Zařízení RouterBOARD je většinou pouze osazený plošný spoj, který je možné umístit do různých krytů podle předpokládaného pracovního prostředí. Většina RouterBOARDů je vybavena sériovým portem RS232, několika ethernetovými (LAN) porty a případně mini PCI slotem. Mini PCI slot slouží k připojení bezdrátového adaptéru s anténou. Takto vybavený RouterBOARD velmi často slouží jako WIFI stanice či přístupový bod (Access Point).

RouterBOARD	CPU [MHz]	RAM [MB]	Počet LAN	Počet mini - PCI	Cena vč. DPH [Kč]
RB 411	300	32	1	1	1017
RB 433	300	64	3	3	2025
RB 711-2Hn	400	32	1	0	931
RB 750	400	32	5	0	893
RB 1100	1066	2000	13	0	8017

Tabulka 3.1: Parametry vybraných RouterBOARDů.

3.3 RouterOS

MikroTik RouterOS je operační systém síťových zařízení RouterBOARD, založený na bázi Linux v2.6 kernel. Tento operační systém je možné nainstalovat i na libovolný stolní počítač se 64MB volného místa na disku. Přestože je RouterOS založen na Linuxu, tedy GNU[2] software, firma MikroTik neposkytuje zdrojové kódy ke svým operačním systémům. V současnosti je k dispozici již sedmá verze tohoto systému (RouterOSv7). Ke každému zakoupenému OS získáme také licenci. Licence se dělí do šesti skupin podle funkcí, možnosti upgradu systému, počtem dnů e-mailové podpory a také podle ceny. V tabulce 3.2 je zobrazen přehled všech šesti licencí RouterOS [8].

Level	**0**	**1**	**3**	**4**	**5**	**6**
Označení	FREE	DEMO	WISP CPE	WISP	WISP	Controller
Cena vč. DPH [Kč]	-	-	-	828	1512	3900
Upgrade na verzi	-	-	ROS v6.x	ROS v6.x	ROS v7.x	ROS v7.x
Podpora (e-mail)	-	-	-	15 dní	30 dní	30 dní
Wireless AP	24h limit	-	-	ANO	ANO	ANO
Wireless Client a Bridge	24h limit	-	ANO	ANO	ANO	ANO
RIP, OSPF, BGP	24h limit	-	ANO	ANO	ANO	ANO
EoIP tunnels	24h limit	1	Neomezeno	Neomezeno	Neomezeno	Neomezeno
PPPoE tunnels	24h limit	1	200	200	500	Neomezeno
PPTP tunnels	24h limit	1	200	200	500	Neomezeno
L2TP tunnels	24h limit	1	200	200	500	Neomezeno
OVPN tunnels	24h limit	1	200	200	Neomezeno	Neomezeno
VLAN interfaces	24h limit	1	Neomezeno	Neomezeno	Neomezeno	Neomezeno
HotSpot active users	24h limit	1	1	200	500	Neomezeno
RADIUS client	24h limit	-	ANO	ANO	ANO	ANO
Queues	24h limit	1	Neomezeno	Neomezeno	Neomezeno	Neomezeno
Web proxy	24h limit	-	ANO	ANO	ANO	ANO
User manager active sessions	24h limit	1	10	20	50	Neomezeno

Tabulka 3.2: Přehled licencí RouterOS.

Ceny uvedené v tabulkách 3.1 a 3.2 jsou převzaty k datu 7. 2. 2012 od oficiálního distributora MikroTik [10] pro Českou republiku.

[2] *GNU* je rekurzivní zkratka (*GNU is Not Unix*) projektu na podporu vývoje svobodného software.

3.4 Možnosti připojení

Abychom mohli RouterBOARD konfigurovat, musíme se k němu nejprve připojit. K těmto zařízením je možné se připojit přes:

- ethernet,
- sériovou linku,
- wifi.

Připojení k RouterBOARDu provádíme pomocí:

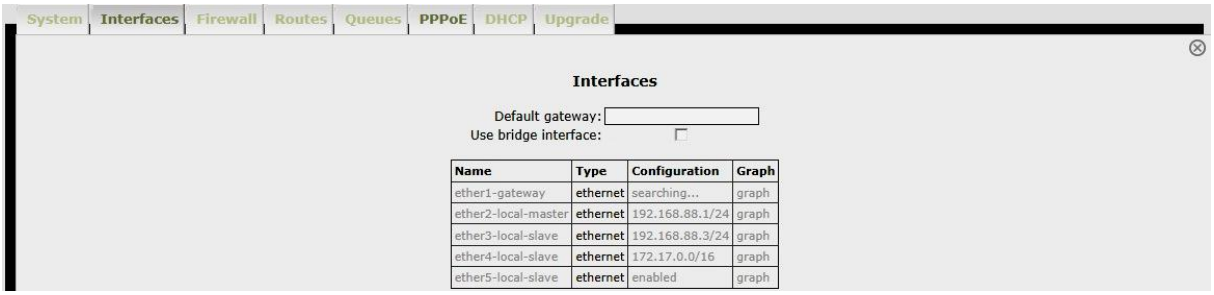
- telnetu/SSH,
- HyperTerminálu,
- webového rozhraní Webbox,
- aplikace Winbox,
- protokolu API.

3.5 Příkazová řádka (telnet, SSH)

Pomocí příkazové řádky je možné konfigurovat všechny funkce RouterBOARDu, ovšem na úkor uživatelského komfortu, který poskytuje grafické rozhraní. Používání příkazové řádky je také podmíněno dobrou znalostí příkazů a struktury RouterOS. K usnadnění konfigurace můžeme použít klávesu „Tab“, která doplní nedopsaný příkaz, obdobně jako u operačního systému Linux. Pomocí příkazu „. .“ opustíme nastavování na dané úrovni a posuneme se o úroveň výše. Ke každému kroku nastavení je možné vyvolat nápovědu napsáním znaku „?“.

3.6 Webové rozhraní Webbox

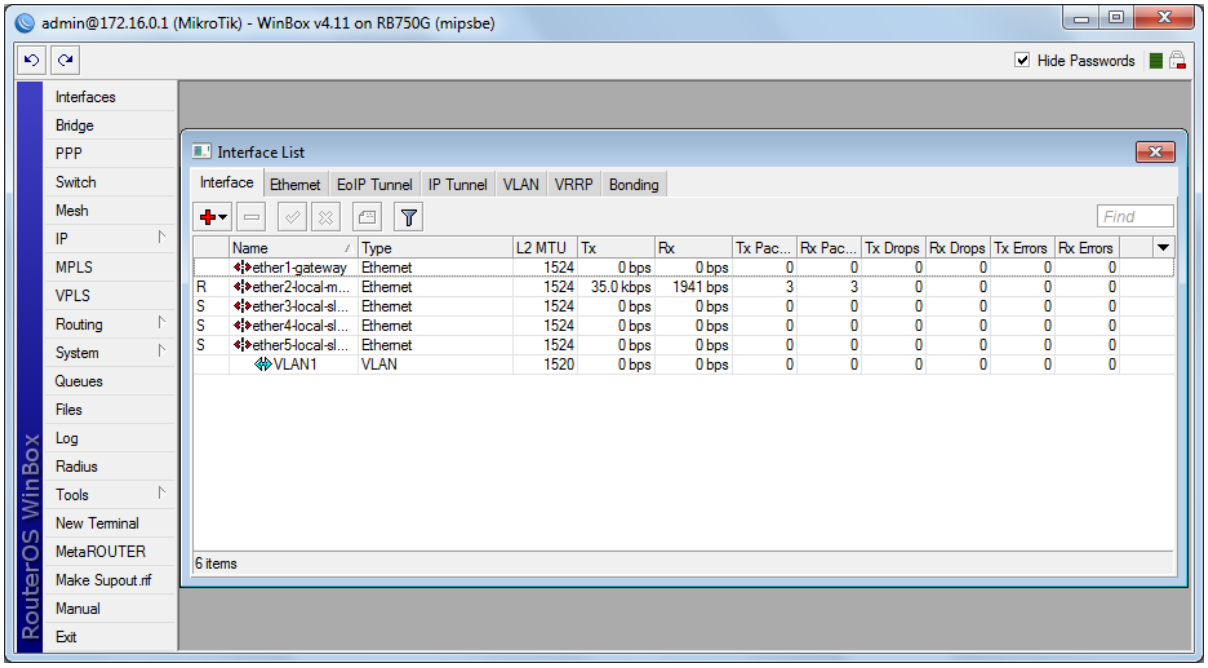
Tak jako naprostá většina směrovačů má i MikroTik webové rozhraní pro konfiguraci zařízení. Toto rozhraní je nazvané Webbox a je určeno pouze k základní správě RouterBOARDU. Zadáním IP adresy připojeného zařízení do webového prohlížeče se vyvolá okno pro vyplnění přihlašovacích údajů. Po úspěšném přihlášení se zobrazí nabídka umožňující provádět základní konfiguraci zařízení jako je např. přidělení IP adresy jednotlivým rozhraním a jejich povolení/zakázání, nastavení NAT či aktivace firewallu. Pokud chceme konfigurovat zařízení s továrním nastavením, připojíme se na IP adresu 192.168.88.1. Toto rozhraní je uživatelsky příjemnější oproti příkazové řádce, neumožňuje však plnohodnotnou konfiguraci zařízení.



Obrázek 3.1: Webové rozhraní Webbox.

3.7 Winbox

Winbox je oficiální program společnosti MikroTik umožňující úplnou konfiguraci zařízení, ale pouze pod operačním systémem MS Windows. Tento program je možné bezplatně stáhnout z webových stránek výrobce. Oproti webovému rozhraní Winbox umožňuje detailnější možnosti nastavení RouterBOARDu. Jedná se například o nastavení dynamického směrování RIP či OSPF, sledování aktuálního vytížení procesoru atd.



Obrázek 3.2: Aplikace Winbox.

Výhodou Winboxu je možnost přihlášení se k zařízení nejen pomocí IP, ale i MAC adresy. Winbox vyhledá všechny dostupné zařízení MikroTik v síti, zobrazí jejich IP resp. MAC adresy a podle volby uživatele se připojí k vybranému RouterBOARDu.

Pro přehlednost a jednoduchost tohoto uživatelského prostředí je Winbox nejčastěji využíván uživateli s OS Windows pro pokročilou konfiguraci RouterBOARDu. Součástí Winboxu je i terminál pro případ, kdy chceme provádět konfiguraci pomocí příkazové řádky.

4 MikroTik API

Tato kapitola se věnuje komunikačnímu protokolu MikroTik API[3], který jsem použil pro komunikaci aplikace s RouterBOARDem. Možnost využít API ke komunikaci se zařízením MikroTik se poprvé objevila u RouterOS verze 3. Protokol API umožňuje programátorům vytvářet aplikace komunikující s RouterBOARDy. Tímto vzniká prostor pro tvorbu nových aplikací pro správu RouterBOARDů.

Protokol API využívá komunikační port 8728. Tento port je u zařízení defaultně zakázán, proto je nutné před prvním použitím tento port nejprve povolit. To můžeme provést například pomocí konzole - příkazem */ip service enable api* nebo pomocí Winboxu v menu IP - Service.

4.1 Komunikační protokol API

Komunikace mezi aplikací a RouterBOARDem je založena na přenosu příkazů a následných odpovědí. Aplikace využívající protokol API odesílá příkaz pro RouterBOARD, který provede příkaz a poté odpoví, zda byl daný příkaz úspěšně vykonán. Pokud aplikace odešle neproveditelný příkaz, tak RouterBOARD odešle zprávu o neúspěšně vykonaném příkazu a chybu konkretizuje.

Příkazy jsou přenášeny ve formě textových řetězců (slov). Každé slovo je zakódováno podle své délky. Tyto řetězce se mohou skládat do vět, které jsou ukončeny slovem nulové délky. Takto zakódované příkazy a odpovědi se odesílají jako pakety TCP[4].

4.1.1 Příkazy

Věta obsahující příkaz je uvozena lomítkem, za kterým následuje příkaz. Jedná se o stejné příkazy jako při zadávání do konzole. Mezery jsou však nahrazeny lomítky viz následující příklady. Počet lomítek tedy koresponduje s úrovní zanoření v nabídce RouterOS.

Příklady příkazů:

`/login`	- přihlášení k RouterOS
`/system/reboot`	- restartování RouterOS
`/system/clock/print`	- výpis systémového času

4.1.2 Argumenty

Při konfiguraci RouterBOARDu si většinou nevystačíme s jednořádkovým příkazem, proto API umožňuje zadávat příkazy složené z více argumentů. Každý z těchto argumentů se odesílá samostatně, přesto se příkaz vykoná jako celek až po obdržení posledního argumentu zakončeného slovem nulové délky.

Každý argument začíná i končí znakem "=", po kterém následuje jeho hodnota. Pomocí těchto argumentů jsme schopni specifikovat všechny parametry daného příkazu. Pořadí argumentů není striktně stanoveno.

---

[3]Aplication Programming Interface – rozhraní pro programování aplikací.

[4]Transmission Control Protocol – základní spojově orientovaný protokol transportní vrstvy, popsaný v RFC 793.

První řádek určuje, jaký příkaz chceme provést. Na dalších řádcích pak následují argumenty. V tomto případě se jedná o vytvoření statického DNS záznamu, argumenty tedy specifikují parametry tohoto záznamu.

Příklad použití argumentů:

```
/ip/dns/static/add
=name=dns1
=address=192.168.88.1
=ttl=1d
```

Pro výpis informací o konfiguraci slouží příkaz *print*. Pokud chceme vypsat pouze určitou část konfigurace, můžeme použít filtr. Ten se tvoří tak, že za příkazem *print* následuje argument započatý znakem „?“. Hodnotou argumentu je filtrovaný text.

Příklad použití filtrování:

```
/interface/print
?type=ether
```

Pomocí tohoto příkazu zobrazíme pouze rozhraní typu ethernet.

4.1.3 Odpovědi

RouterOS odpovídá po provedení příkazu zprávou začínající znakem „!“. RouterOS rozlišuje následující tři typy odpovědí:

- `!re` - označuje začátek odpovědi a následuje vlastní odpověď
- `!done` - značí úspěšně vykonaný příkaz
- `!trap` - značí neúspěšně vykonaný příkaz, dále následuje popis chyby

4.2 Dostupnost API

Na webových stránkách MikroTiku věnovaných popisu API [6] jsou umístěny příklady zdrojových kódů pro komunikaci s RouterOS. Zveřejněné zdrojové kódy jsou vytvořeny v různých programovacích jazycích např. C, C++, C#, Delphi, Java, Perl, PHP a Python.

5 Návrh aplikace

Hlavní částí této kapitoly je znázornění struktury navržené aplikace, z které jsou zřejmé všechny funkce, které aplikace podporuje. V druhé části kapitoly jsem provedl teoretický rozbor těchto funkcí. Poslední část popisuje grafickou podobu navržené aplikace.

5.1 Struktura navržené aplikace

Při výběru funkcí, které bude aplikace podporovat, jsem vycházel ze zadání bakalářské práce. To znamená, že zpočátku aplikace podporovala pouze tyto tři funkce: DHCP, DNS a statický routing.
Po konzultaci s vedoucím bakalářské práce jsem doplnil funkce o dynamické směrovací protokoly RIP a OSPF.

Aby aplikace alespoň částečně zastoupila Winbox při konfiguraci RouterBOARDu pod operačním systémem Linux, pro který není Winbox určen, bylo nutné implementovat další funkce. Aplikace se tedy rozrostla o řízení datových toků Queue a Firewall. Takto navržená aplikace již umožňovala rozsáhlejší správu zařízení než Webbox. Přesto jsem se rozhodl pokračovat v rozšiřování funkcí aplikace. Výsledkem je podpora správy uživatelů a jejich oprávnění při konfiguraci RouterBOARDu, možnost zálohování konfigurace RouterBOARDu, konfigurace protokolu ARP, VLAN a další.

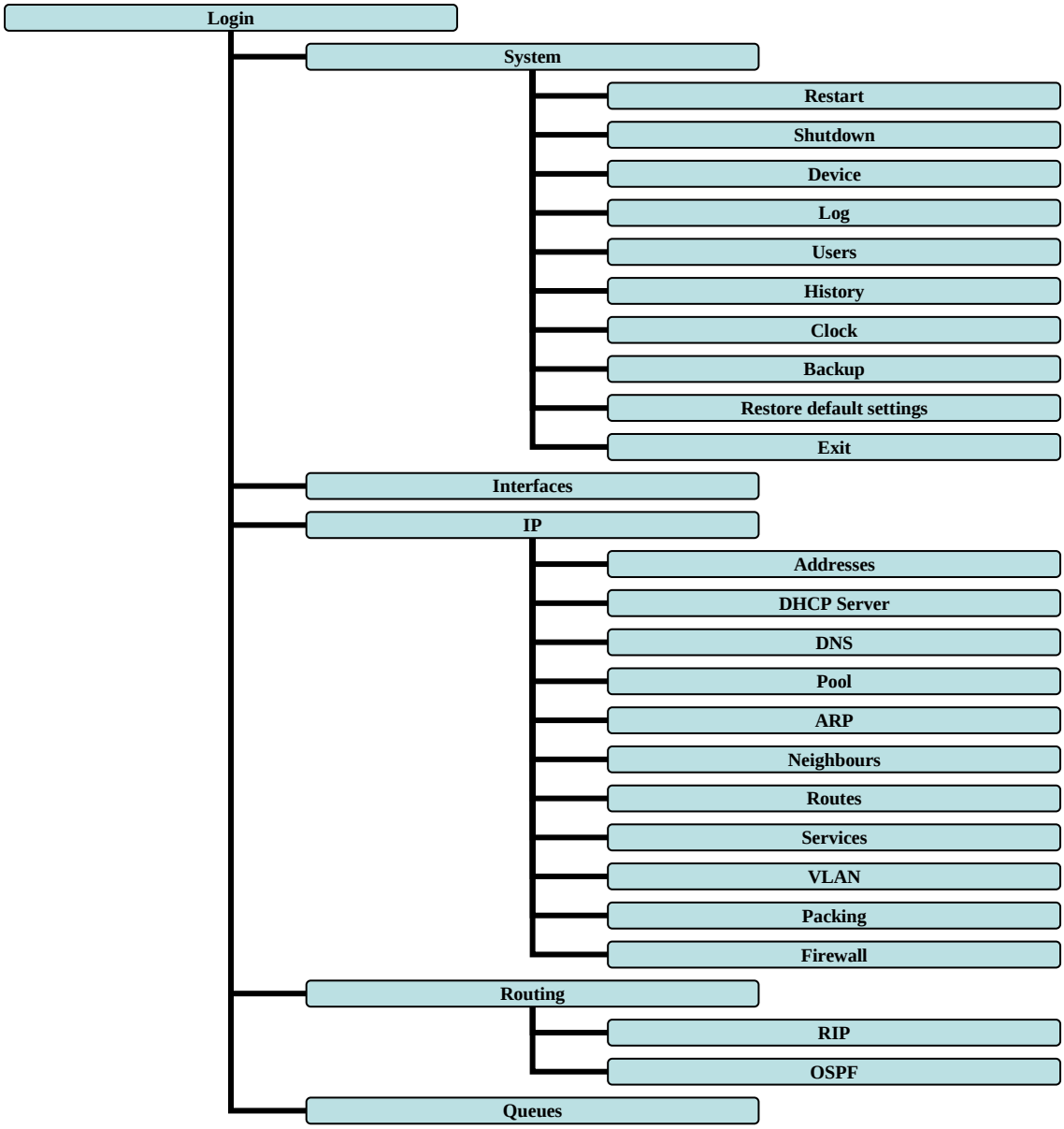
Takto je již podporována dostatečně široká škála funkcí a tím je umožněna správa většiny funkcí, které Winbox respektive RouterOS podporuje.
Konečná struktura navržené aplikace je znázorněna pomocí blokového schématu na obrázku 5.1.

5.2 Teoretický rozbor implementovaných funkcí

Jelikož aplikace obsahuje mnoho odborných názvů z oblasti počítačových sítí, rozhodl jsem se tyto termíny do češtiny nepřekládat. Překlad by mohl být pro uživatele matoucí.
Jak je zřejmé z obrázku 5.1, implementované funkce jsou rozděleny do následujících kategorií:

- System,
- Interfaces,
- IP,
- Routing,
- Queues.

V následujícím textu jsou vysvětleny všechny funkce, které jsou implementovány v jednotlivých kategoriích. Při psaní této podkapitoly jsem čerpal informace z odborné literatury [1,2,9].



Obrázek 5.1: Struktura aplikace.

5.2.1 Kategorie System

Tato kategorie obsahuje základní nástroje pro obsluhu systémových služeb RouterBOARDu.

- **Reboot** – volbou této položky se provede restartování RouterBOARDu. Stejně jako u Winboxu má restart RouterBOARDu za následek restartování i celé aplikace.
- **Shutdown** – provede vypnutí RouterBOARDu i aplikace.
- **Device** – vypíše systémové informace o RouterBOARDu jako je frekvence CPU, vytížení CPU, celková a volná paměť, název modelu, atd.
- **Log** – slouží k vedení záznamů o uživatelích, kteří se připojili k RouterBOARDu. Eviduje se čas připojení, uživatelské jméno, IP adresa uživatele a rozhraní, přes které se uživatel připojil.
- **Historie** – eviduje všechny provedené změny v konfiguraci RouterBOARDu. Každý záznam obsahuje jméno uživatele, který provedl změny, čas a datum této změny.

- **Users** – umožňuje vytvářet nové uživatele a skupiny. Každému uživateli lze nastavit přihlašovací heslo, skupinu a IP adresu, ze které se může daný uživatel přihlásit k RouterBOARDu. Každé skupině je možno nastavit různá práva jako čtení, zápis, možnost použití Winboxu nebo API či provést restart RouterBOARDu.
- **Clock** – slouží k nastavení data a času RouterBOARDu.
- **Backup** – umožňuje vytvořit zálohu konfigurace RouterBOARDu, tato konfigurace je uložena v paměti RouterBOARDu. Pomocí této zálohy je možné kdykoli zpětně obnovit konfiguraci RouterBOARDu. Záloha ve svém názvu obsahuje datum a čas vytvoření.
- **Restore default settings** – vyvolá tovární nastavení RouterBOARDu. Ekvivalentem je HW restart RouterBOARDu.
- **Exit** – je poslední položkou v kategorii System, která ukončí aplikaci.

5.2.2 Kategorie Interfaces

Kategorie interfaces je dostupná z hlavního okna po přihlášení dvojklikem na dané rozhraní. Umožňuje konfiguraci jednotlivých rozhraní typu ethernet nebo wireless. Podrobný popis konfigurace je uveden v podkapitole 5.3.1.

5.2.3 Kategorie IP

Tato kategorie obsahuje nástroje, pomocí kterých je možné provést pokročilé síťové nastavení.

- **Addresses** – nabídka pro konfiguraci IP adres. Jednotlivým rozhraním je možné nastavit IP adresu, masku, adresu sítě a broadcast. V současné době se stále ještě nejčastěji využívá IPv4, která se skládá z 32bitové adresy, zapsané dekadicky po jednotlivých oktetech oddělených tečkou. Z důvodu nedostatku IP adres bude protokol IPv4 nahrazen 128bitovým protokolem IPv6.
- **DHCP Server** (Dynamic Host Configuration Protocol) – slouží k automatické konfiguraci stanic v síti. Klientské stanice žádají DHCP server o přidělení IP adresy, síťové masky a výchozí brány. Při konfiguraci služby DHCP Server můžeme pro přidělení IP adresy zvolit možnost statického přidělení IP adresy nebo dynamického přidělení IP adresy z rozsahu IP Pool. Samozřejmostí je možnost volby Lease Time, která udává, jak dlouho může byt daná IP adresa vypůjčena.
- **DNS** (Domain Name Server) – každý počítač v počítačové sítí má pro identifikaci svou jedinečnou IP adresu. DNS umožňuje každé IP adrese přiřadit tzv. doménové jméno, které je pro uživatele snáze zapamatovatelné. Když do webového prohlížeče napíšeme adresu v doménovém tvaru, počítač se zeptá DNS serveru jaká IP adresa přísluší zadanému doménovému jménu a pak se připojí na výslednou (přeloženou) IP adresu. Kromě adresy DNS serveru je v této nabídce umožněno vytvořit statické záznamy, které budou použity v případě dočasné nedostupnosti DNS [2].
- **Pool** – definuje rozsah IP adres, z kterého může být IP adresa serverem DHCP přidělena. Rozsah je dán počáteční a koncovou IP adresou.

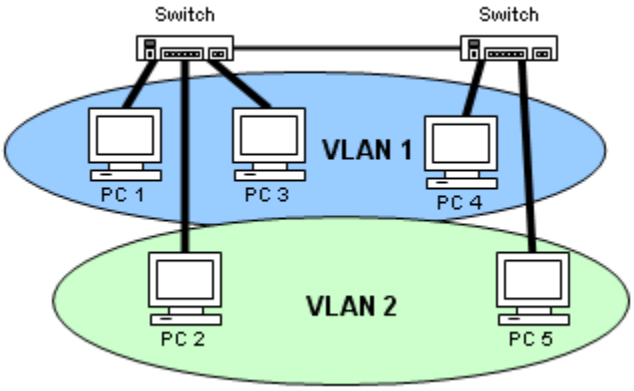
- **ARP** (Address Resolution Protocol) – umožňuje překlad IP adresy na fyzickou (MAC) adresu. ARP využívá všesměrové vysílání (broadcast) pro zjištění MAC adresy vlastníka cílové IP adresy. Pokud je cílová stanice připojena na stejný segment sítě odpoví na dotaz svou fyzickou adresou. Pokud se ovšem cílová adresa nachází v jiném síťovém segmentu, odpoví fyzickou adresou směrovač, který zná cestu do příslušné sítě.
- **Neighbors** – RoterBOARD používá MNDP (MikroTik Neighbor Discovery Protocol), který slouží k nalezení připojených zařízení k broadcastové doméně kompatibilních s MNDP nebo CDP (Cisco Discovery Protocol). Ke každému nalezenému zařízení se vypisuje rozhraní, ke kterému je toto zařízení připojeno, IP adresa, MAC adresa a název zařízení.
- **Routes** – tato funkce umožňuje nastavení statického směrování (routování). Při statickém směrování se využívá jedna předem definovaná cesta k cíli, která je obvykle nakonfigurována správcem sítě. V případě, že vypadne některý ze spojů nebo směrovačů na dané cestě, není možnost dynamického přesměrování paketu a tak tento paket nemůže být doručen k cíli. Statická cesta se nejčastěji používá v případě, kdy existuje pouze jedna cesta k cílové síti, proto by bylo zbytečné, aby směrovač zjišťoval alternativní cesty k cíli. Dále se statické cesty používají z důvodu bezpečnosti v případech, kdy je nutné, aby byla použita předem známá a bezpečná cesta sítí. Statické směrování je možné kombinovat se směrováním dynamickým. Obvykle má statická cesta prioritu před cestou dynamickou. Pokud se navolí priorita dynamické cesty, je cesta statická použita pouze při selhání dynamického směrovacího protokolu [1]. V tabulce 5.1 jsou porovnány vlastnosti obou druhů směrování.

Vlastnost	Statické směrování	Dynamické směrování
Reakce na změny topologie sítě	NE	ANO
Možnost rozdělení zátěže mezi více cest	Pouze staticky	ANO
Potřeba správce pro manuální konfiguraci	Vysoká	Malá
Dohled nad používanými cestami	Vysoký	Malý
Výměna směrovacích informací	Žádná	ANO
Zátěž směrovače	Minimální	Střední až vysoká
Zátěž paměti	Minimální	Střední až vysoká
Zátěž sítě	Žádná	Střední

Tabulka 5.1: Porovnání statického a dynamického směrování [1].

- **Services** - tato nabídka slouží pro povolení či zakázání služeb a jejich portů na kterých RouterBOARD naslouchá. Spravovat můžeme ty služby:
    - API,
    - FTP,
    - SSH,
    - telnet,
    - Winbox,
    - WWW,
    - WWW-SSL.

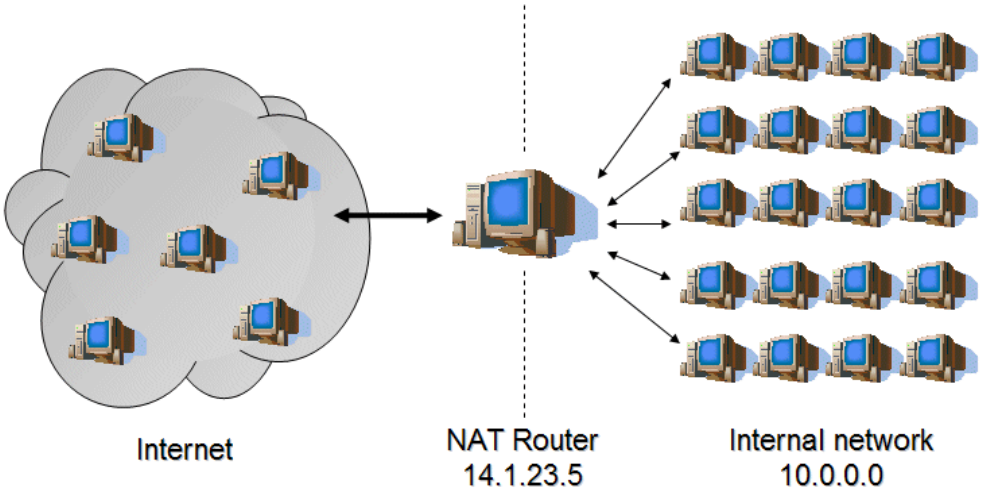
- **VLAN** (Virtual Local Area Network) - umožňuje seskupit pracovní stanice do jedné virtuální lokální sítě (VLAN) bez ohledu na fyzické umístění stanic (viz obrázek 5.2). Stanice se tedy mohou nacházet v různých segmentech sítě, a přesto spolu mohou komunikovat jako by se nacházely v jednom společném segmentu sítě. Základním prvkem pro tvorbu VLAN je přepínač (switch). Na každém přepínači je možné definovat, které porty budou spadat do společné VLAN. Takto vytvořená VLAN je limitována tím, že každá stanice může být členem právě jedné VLAN. Pokud bychom chtěli, aby jedna stanice spadala do více VLAN museli bychom členství ve virtuálních sítích definovat pomocí MAC adres. VLAN jsou obrovským přínosem v logickém členění sítě na skupiny. Z hlediska bezpečnosti počítačové sítě je možné každé skupině nastavit odlišná práva (např. pro přístup na internet) a zároveň izolovat pracovní stanice v rámci pracovních skupin.



Obrázek 5.2: Znázornění VLAN [12].

- **Packing** – umožňuje nastavit kompresi a agregaci paketů, abychom dosáhli vyšší propustnosti zařízení.
- **Firewall** – je služba, která slouží k ochraně privátní sítě před hrozbami z veřejné sítě – Internetu. Firewall odděluje provoz mezi dvěma sítěmi, přičemž propouští příchozí i odchozí data podle předem daných pravidel. Pakety jsou tedy na základě těchto pravidel přijaty nebo zahozeny. V praxi se velmi často setkáváme s kombinací hardwarových i softwarových firewallů. V nabídce Firewall je obsažena i funkce NAT (Network Address Translation),

která umožňuje překládat adresy z privátního rozsahu adres do veřejného. NAT vznikl z důvodu omezeného počtu IP adres. Jak je znázorněno na obrázku 5.3, pomocí NAT je možné „skrýt“ celou privátní síť za jedinou veřejnou IP adresu, která komunikuje s vnější sítí (Internetem). Princip NATu je takový, že všechny počítače v privátní síti mají nastavenu jako výchozí bránu adresu routeru, který je připojen k Internetu. Paket z každého počítače, který směřuje mimo privátní síť je předán routeru, který v hlavičce paketu nahradí zdrojovou adresu za adresu vlastní. Takto upravený paket, je poté odeslán do sítě Internetu. Ve chvíli, kdy cílový počítač odpovídá na přijatý paket, odešle paket zpět na adresu routeru. Router přepíše cílovou adresu přijatého paketu na adresu původního odesilatele z privátní sítě a paket doručí. Další výhodou NATu je, že zvyšuje bezpečnost počítačů v privátní síti, jelikož potenciální útočník z Internetu nezná skutečnou adresu počítače.



Obrázek 5.3: Znázornění NAT [13].

5.2.4 Kategorie Routing

Tato kategorie obsahuje nástroje pro konfiguraci dynamických směrovacích protokolů.

Dynamické směrování průběžně reaguje na změny v počítačové síti a těmto změnám přizpůsobuje obsah směrovací tabulky. Pro výběr nejlepší cesty do cílové sítě se používá algoritmus, který využívá aktuální směrovací informace, které směrovač obdrží od ostatních směrovačů. Výměna těchto informací je dána směrovacím protokolem. Podle typu směrovacího protokolu se rozesílají aktuální směrovací informace periodicky nebo v případě detekování změn síťové topologie.

Každý dynamický směrovací protokol se snaží nalézt optimální cestu k cíli. Pro nalezení optimální cesty se využívá jednoho nebo více kritérií. Za optimální cestu je považována ta, která nejlépe splní daná kritéria. Tyto kritéria se souhrnně nazývají metrikou.

Metrikou obvykle bývá [1]:

- Počet směrovačů na dané cestě – podle tohoto kritéria nejnižší počet směrovačů na cestě znamená nejlepší cestu.
- Propustnost přenosového média – podle tohoto kritéria je nejlepší cesta dána nejvyšším součtem propustností všech spojů.
- Zpoždění – cesta s nejnižším zpožděním při průchodu přes všechny spoje na cestě je podle tohoto kritéria nejlepší.

- Spolehlivost – spolehlivostí se rozumí pravděpodobnost toho, že data budou doručena. Podle tohoto kritéria je nejlepší ta cesta, která má nejvyšší spolehlivost.
- Zátěž – zátěž se určuje podle procentuálního vytížení přenosového média. Nejlepší cestou je cesta s nejnižší zátěží.

Do kategorie Routing jsem zařadil následující dva dynamické směrovací protokoly:

- **RIP** (Routing Information Protocol) – je jedním z prvních úspěšných směrovacích protokolů. Byl vyvinut firmou Xerox v roce 1981. První normalizovaná verze tohoto protokolu je popsána v RFC[5] 1058, vylepšená verze označovaná jako RIPv2 je z roku 1994 a je popsána v RFC 1723. RIP využívá jako metriku počet směrovačů na cestě k cíli. Metrikou 0 je ohodnocena síť, která je připojená přímo ke směrovači. Nejvyšší platná hodnota metriky může být 15, což znamená, že RIP je limitován 15 směrovači na cestě k cíli. Metrika 16 se používá pro označení neplatných cest.

  Směrovací tabulka obsahuje následující údaje [1]:

  - cílová adresa,
  - metrika,
  - adresa nejbližšího směrovače na cestě k cíli,
  - doba od poslední aktualizace záznamu.

  RIP si vyměňuje směrovací informace (tabulky) pouze se svými nejbližšími sousedy. Směrovací informace jsou odesílány na všesměrovou adresu (broadcast) každých 30 sekund. Po každé výměně směrovací tabulky směrovač připočte k metrice každé cesty jedničku a porovná tuto hodnotu se svým stávajícím záznamem. Pokud je metrika nižší, směrovač přepíše stávající záznam. Jinak je nová informace ignorována. Pro přenos tabulek se využívá transportní protokol UDP. V následující tabulce jsou shrnuty výhody a nevýhody směrovacího protokolu RIP.

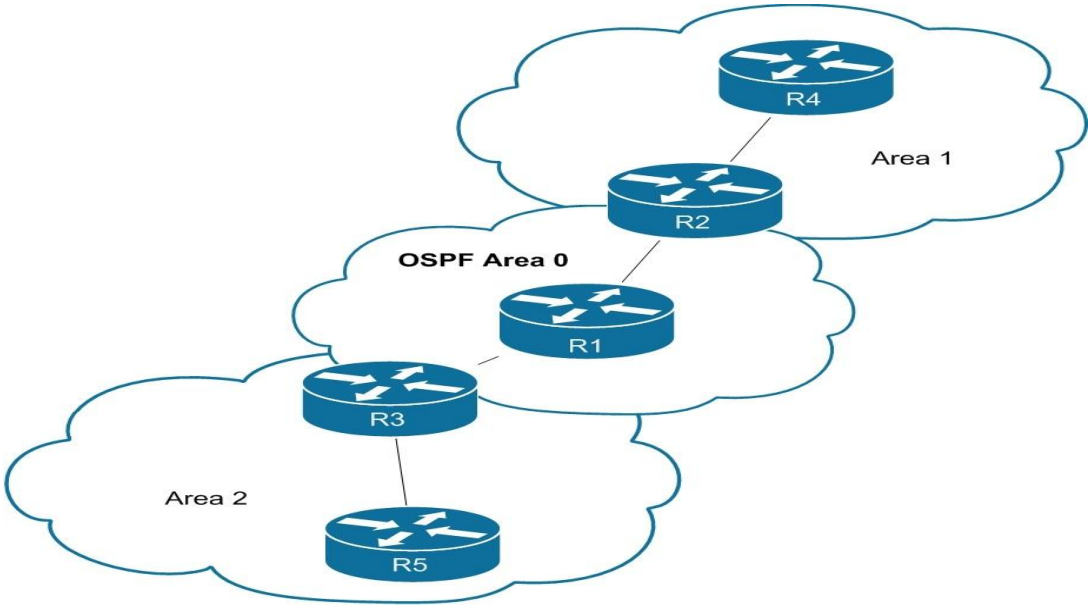
Výhody	Nevýhody
Otevřený směrovací protokol	Jednoduchá metrika
Jednoduchá konfigurace	Pomalá konvergence
	Nejdelší cesta 15 skoků
	Není možné rozdělit zátěž do paralelních cest

Tabulka 5.2: Výhody a nevýhody protokolu RIP [1].

[5]Request For Comments - označení řady standardů popisujících internetové protokoly.

- **OSPF** (Open Shortest Path First) – směrovací protokol OSPF je navržen pro rozsáhlé počítačové sítě. Byl vyvinut v letech 1988 až 1991 a je popsán v RFC 1131. OSPF verze 2 z roku 1998 je popsána v RFC 2328. Tento algoritmus hledá nejkratší cestu mezi směrovačem a dostupnými sítěmi pomocí algoritmu SPF (Shortest Path First). Velkou výhodou je jeho efektivita. Proto ani ve velmi rozsáhlých sítích nedochází k výraznějšímu zatížení přenosových cest.

  Směrovače si nevyměňují směrovací tabulky jako při použití protokolu RIP, ale udržují si mapu struktury propojených sítí, kterou aktualizují při každé změně topologie sítě nebo minimálně každých 30 minut. Tato mapa se odborně nazývá topologická databáze sítě. OSPF využívá hierarchické směrování, kdy se sítě a směrovače v jednom autonomním systému dělí do tzv. oblastí (area). Tyto oblasti jsou dále spojovány hraničními směrovači.



Obrázek 5.4: Znázornění OSPF [16].

Znalost topologie dané oblasti zůstává ostatním oblastem skryta, čímž je umožněno provádět změny topologie v každé oblasti nezávisle na ostatních oblastech. Páteř OSPF (backbone) je tvořena právě jednou oblastí ke které musí být připojeny všechny ostatní oblasti.

Výpočet algoritmu SPF pro výpočet nejkratších cest se spouští samostatně pro každou oblast. Proto změna topologie sítě v jedné z oblastí nevyvolá přepočet SPF algoritmu v ostatních oblastech. Hraniční směrovače oblastí mají tolik databází, kolik oblastí propojují. Směrování pomocí OSPF je založeno na bezrozměrné metrice, označované jako cena, což je číslo v rozsahu 1 až 65535 přiřazené ke každému rozhraní směrovače. Nižší číslo znamená lepší metriku. Existuje-li více cest se stejnou cenou, je možné rozdělit zatížení mezi tyto cesty. Stejně jako u předchozího protokolu jsou v následující tabulce shrnuty výhody a nevýhody tohoto protokolu.

Výhody	Nevýhody
Otevřený směrovací protokol	Složitý návrh sítě
Rychlá konvergence	Kvalita směrování v závislosti na použité adresaci
Vysílání směrovacích informací na skupinovou adresu	
Podpora paralelních cest	
Podpora VLSM a CIDR	

Tabulka 5.3: Výhody a nevýhody protokolu OSPF [1].

5.2.5 Kategorie Queues

Tato kategorie obsahuje nástroj pro řízení datového toku Queue. Jímž můžeme zajistit kvalitu služeb (QoS). Pomocí tohoto nástroje jsme schopni omezovat či upřednostňovat rychlost downloadu, ale i uploadu jednotlivých IP adres.

RouterOS umožňuje použití několika mechanismů řízení síťového provozu [9]:

- **PFIFO** (Packets First In First-Out) – tento mechanismus řadí pakety do fronty v pořadí, v jakém příjdou. Použití této fronty není příliš vhodné pro omezování rychlostí, dochází zde k zahlcování sítě a při zaplnění fronty jsou nově příchozí pakety zahazovány. Tento mechanismus je defaultně nastaven v RouterOS.
- **SFQ** (Stochastic Fair Queueing) – využívá jednoduchý algoritmus pro zajištění rovnoměrného vytížení sítě s velkými datovými přenosy. SFQ není zaměřeno na uživatele nebo programy, ale na jednotlivé TCP a UDP přenosy dat. Vytváří se zde více front, do kterých jsou řazeny TCP spojení a UDP proudy. Tyto fronty jsou následně obsluhovány algoritmem Round Robin. Pakety jsou tedy rovnoměrně odebírány, čímž je umožněn přístup k přenosovému pásmu každému datovému toku. Z tohoto důvodu je tato metoda doporučována pro přetížené sítě.
- **PCQ** (Per Connection Queue) – je podobné metodě SFQ, umožňuje ovšem vytvářet další podfronty, které je možné klasifikovat podle adresy zdrojové, cílové nebo čísla portu.
- **RED** (Random Early Detection) – je velmi často využívaný mechanismus, protože dokáže ohleduplně řídit vytížení sítě. Tato metoda detekuje stav, kdy se blíží zahlcení sítě a na tento stav reaguje náhodným zahazováním paketů čekajících ve frontě. To znamená, že čím více je linka zahlcená, tím více paketů bude zahozeno. Tento mechanismus je ovšem možné použít pouze na TCP spojení.

5.3 Grafický návrh uživatelského rozhraní

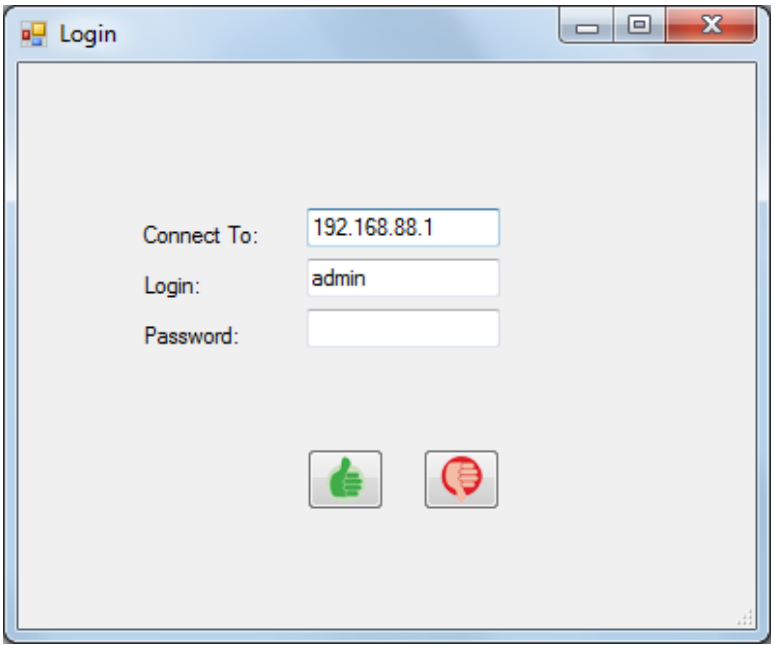
V této části kapitoly je prezentována grafická koncepce aplikace spolu s ovládáním uživatelského prostředí a objasněním významu použitých grafických prvků.

5.3.1 Uživatelské rozhraní

Aplikaci jsem navrhl tak, aby její obsluhování bylo snadné a intuitivní. Proto se jedná o aplikaci s grafickým rozhraním, kde se všechny funkce nastavují pomocí jednoduchých formulářů.

Aplikace se skládá ze 49 formulářů (oken), z toho jeden formulář představuje hlavní okno aplikace. Zbylé formuláře slouží pro správu funkcí, které aplikace podporuje. Z důvodu dosažení přehlednosti jsou okna členěna pomocí záložek.

Po spuštění aplikace se zobrazí úvodní obrazovka pro přihlášení k RouterBOARDu. Aby bylo možné se připojit, musíme znát IP adresu zařízení a mít platné přihlašovací jméno i heslo. Heslo je nepovinné, jelikož RouterOS umožňuje vytvořit uživatele, který nemá žádné heslo.



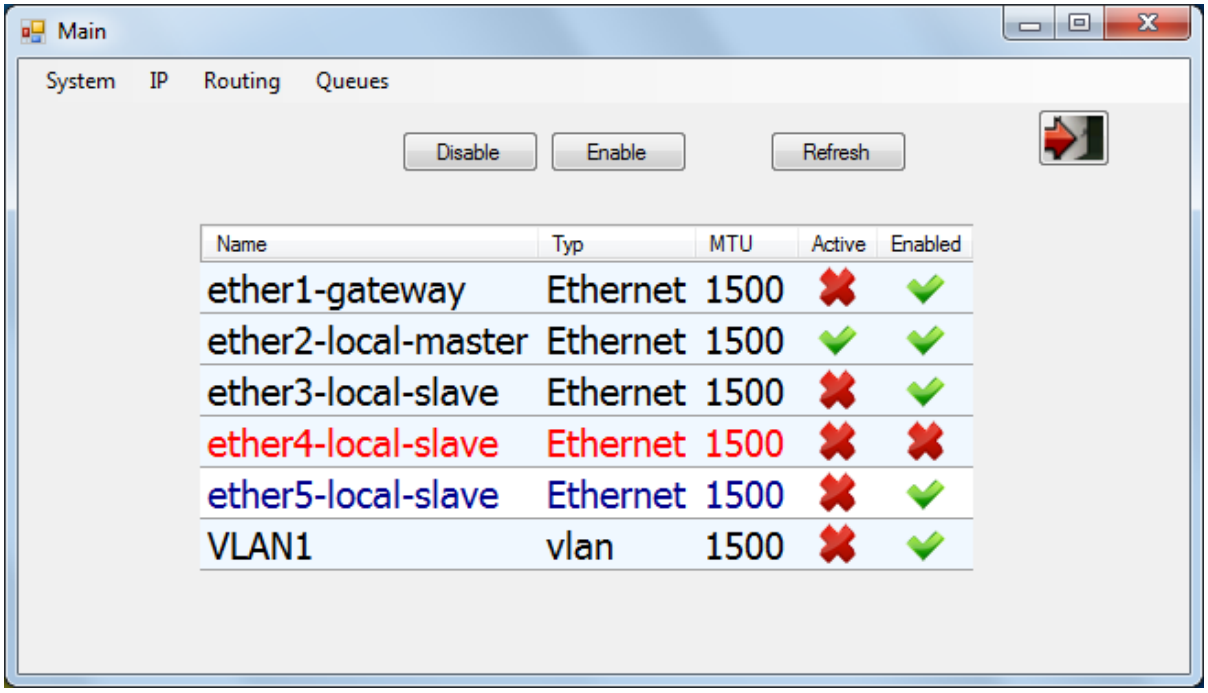
Obrázek 5.5: Úvodní obrazovka pro přihlášení do aplikace.

RouterBOARD s továrním nastavením má IP adresu nastavenu na: 192.168.88.1. Jediným registrovaným uživatelem je admin, který nemá žádné heslo. Uživatele je samozřejmě možné kdykoli po přihlášení editovat i přidávat nové. Pokud neznáme přihlašovací údaje, není možné se k RouterBOARDu připojit. Proto je v případě zapomenutého hesla jedinou možností RouterBOARD fyzicky resetovat do továrního nastavení, k čemuž slouží resetovací tlačítko. Reset RouterBOARDu je poněkud nestandartní. Pokud chceme RouterBOARD resetovat, musíme nejprve vypnout napájení. Poté stisknout a stále držet resetovací tlačítko a připojit napájení. Tlačítko necháme stisknuté, dokud LED dioda nepřejde ze stavu svícení do stavu blikání.

Po úspěšném přihlášení se zobrazí hlavní okno aplikace. V horní části okna se nachází hlavní menu a následuje výpis všech rozhraní. Automaticky jsou zde zobrazeny všechny rozpoznané síťové adaptéry (ethernet i wifi) a také virtuální adaptéry (VLAN i bridge).

Pro výpis požadovaných informací je v aplikaci nejčastěji použit *dataGridView*, který zobrazuje informace v přehledné tabulce. Po kliknutí na řádek, který chceme nastavovat, se změní barva pozadí i písma tohoto řádku, aby bylo zřejmé, který řádek právě nastavujeme.

Ke každému rozhraní jsou pomocí zelených zatržítek a červených křížků zobrazeny informace, zda je rozhraní aktivní/neaktivní či povoleno/zakázáno. Tyto stavy lze nastavovat pomocí tlačítek Disable a Enable. Text v řádcích u zakázaných rozhraní je z důvodu přehlednosti napsán červenou barvou. Tlačítko Refresh slouží k aktualizaci zobrazeného výpisu.



Obrázek 5.6: Hlavní obrazovka.

Dvojklikem na konkrétní rozhraní se zobrazí okno pro konfiguraci zvoleného rozhraní. U každého rozhraní je vypsána MAC adresa a je možné měnit jeho název, nastavovat velikost MTU[6] či ARP režim. Ethernetovému rozhraní je možné nastavit jeho rychlost (10/100/1000 Mbps), half/full duplex a auto negotiation. U bezdrátových rozhraní lze nastavit např. SSID[7] nebo pracovní frekvenci.

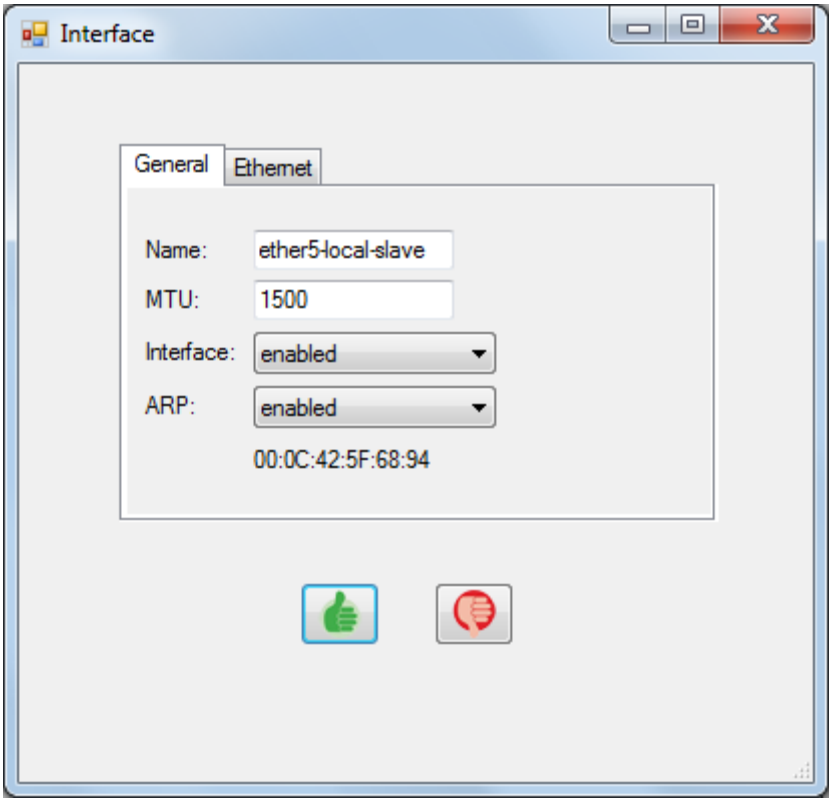
Pro přístup ke správě dalších podporovaných funkcí slouží hlavní menu. Vybráním položky z menu se vytvoří nové okno. Nově vytvořené okno se zobrazí modálně, to znamená, že uživatel musí před výběrem další položky z menu nejprve ukončit právě otevřené okno. Hlavní okno tedy zůstává zobrazené po celou dobu běhu aplikace.

Vypsaná data se obnovují při otevírání nových oken či při přepínání záložek a dále se data automaticky obnoví po provedení libovolných změn konfigurace uživatelem. Výjimkou jsou funkce, kdy je nutná okamžitá aktualizace dat, jako například stav vytížení procesoru RouterBOARDu vyjádřený v procentech. V takovýchto případech se vypsaná data aktualizují periodicky.

[6]Maximum Transmission Unit – maximální velikost paketu.

[7]Service Set Identifier – jedinečný identifikátor bezdrátové sítě.

V celé aplikaci jsou použita tlačítka s obrázky, jak je ukázáno na příkladu okna nastavení rozhraní (viz obrázek 5.7). Zelené tlačítko slouží pro vykonání akce a červené tlačítko naopak slouží pro zavření formuláře bez provedení změn.



Obrázek 5.7: Konfigurace rozhraní.

Aplikace kontroluje data vyplněná uživatelem při požadavku potvrzení změn, pokud chybí nebo je špatně zadán některý z povinných parametrů, upozorní uživatele na tuto skutečnost. V opačném případě aplikace odešle příkaz pro RouterBOARD. Může se stát, že uživatel vyplní vše potřebné pro odeslání příkazu, ale přesto není možné příkaz vykonat. To bývá zapříčiněno nevhodně vyplněnými daty např. zadaná IP adresa, která se nenachází v síťovém segmentu daném maskou sítě. Na takový příkaz RouterBOARD reaguje odesláním chybové zprávy. Tato zpráva se poté zobrazí uživateli na obrazovce. Uživatel pak může opravit vstupní data nebo ukončit nastavování.

K odhlášení z aplikace slouží tlačítko v pravé horní části hlavního okna. Po odhlášení se automaticky zobrazí přihlašovací okno pro nové přihlášení. Ukončit celou aplikaci je možné v menu System – položka Exit.

6 Implementace

Tato kapitola je rozdělena do dvou částí. V první části jsou popsány nástroje a použité technologie, v druhé části je vysvětlena implementace. Informace uvedené v této kapitole jsou čerpány z odborných knih [3,4] zabývajících se touto problematikou.

6.1 Použité technologie

Při tvorbě aplikace jsem se rozhodl použít programovací jazyk C# spolu se softwarovou platformou .NET Framework. Implementace byla prováděna ve vývojovém prostředí Microsoft Visual Studio 2010 Professional. Pro přenositelnost aplikace mezi operačními systémy Microsoft Windows a Linux jsem použil framework Mono.

6.1.1 .NET Framework

V dnešní době jsou stále častěji vytvářeny složité aplikace, které svým rozsahem převyšují možnosti jednoho člověka. Proto se na vývoji rozsáhlých aplikací většinou podílí více programátorů. Často se používají již vytvořené komponenty, tehdy mluvíme o modulárním programování, které spočívá ve vytvoření programového celku z menších úseků – podprogramů (funkcí a procedur).

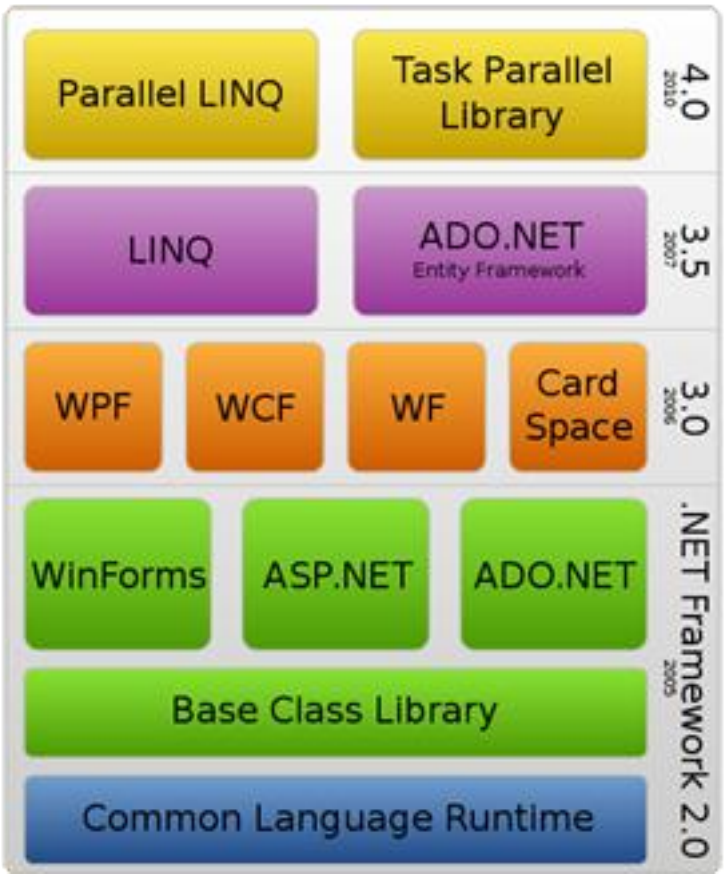
Snahou této platformy bylo zjednodušit a sjednotit vývoj aplikací. Cílem bylo, aby .NET aplikace mohli vytvářet všichni programátoři, bez ohledu na použitý programovací jazyk. Jednou z výhod platformy .NET je velké množství různých druhů projektů, které lze pomocí tohoto frameworku vyvíjet. S použitím .NET jsme schopni vytvářet nejen klasické konzolové aplikace nebo uživatelsky příjemné formulářové aplikace, ale i webové aplikace či zásuvné moduly Microsoft Office. Formulářové aplikace založené na Windows Forms využívají volání Win32 API. K vývoji webových aplikací je možné použít ASP.NET WebForms.

.NET Framework existuje v pěti verzích: 1.0, 1.1, 2.0, 3.0, 3.5, 4.0. Aktuální je verze 4.0, která byla představena roku 2010. V současné době je firmou Microsoft podporováno pět různých programovacích jazyků, kterými jsou C++, C#, Visual Basic, F# a JavaScript. Tyto programovací jazyky jsou integrovány do jediného vývojového prostředí MS Visual Studio.

Výhody použití .NET Frameworku:

- zjednodušuje vývoj a nasazení aplikace,
- robustní a bezpečné prostředí pro běh aplikace,
- rozsáhlé knihovny funkcí,
- multijazyková podpora.

Architektura .NET Frameworku sestává z komponent zobrazených na obrázku 6.1. V anglické literatuře se často setkáme s názvem .NET Framework stack.



Obrázek 6.1: Architektura .NET [11].

Základními komponentami .NET Frameworku jsou:

- **Common Language Runtime** (CLR) – exekuční prostředí pro běh .NET Framework aplikací. Umožňuje spuštění kódu, správu paměti, zpracování výjimek a konverzi mezikódu na nativní kód platformy. Zdrojové kódy nejsou kompilovány přímo do nativního kódu, ale do intermediárního jazyka (podobně jako v programovacím jazyce Java).
- **Base Class Library** (BCL) – je základní knihovna, kterou využívají všechny programovací jazyky .NET. Poskytuje rozsáhlou množinu tříd, které umožňují přístup k interním vlastnostem operačního systému.
- **Windows Forms** – slouží pro vývoj okenních aplikací.
- **ASP.NET** – slouží pro vývoj dynamických a výkonných webových aplikací a služeb.
- **ADO.NET** – představuje množinu tříd nabízejících služby pro přístup k datům a tvorbu databázových aplikací.

6.1.2 Programovací jazyk C#

Programovací jazyk C# vyvinula společnost Microsoft. Jak již název napovídá, tento jazyk vychází z programovacího jazyka C/C++, ovšem v mnoha ohledech se více podobá programovacímu jazyku Java. Jazyk C# je určen pro vývoj aplikací založených na .NET Frameworku.

Představen byl společně s .NET Frameworkem 1.0 v roce 2002. Tato verze již obsahovala základní podporu objektového programování. Roku 2005 vyšla nová verze C# 2.0, která nabídla nové doplňky jazyka, jako jsou statické třídy, generika a iterátory. Koncem roku 2007 vznikla verze C# 3.0. Hlavními novinkami oproti předchozím verzím byly LINQ (integrovaný dotazovací jazyk) a lambda výrazy. Nová verze C# 4.0 vyšla roku 2010 a nově podporuje dynamicky typované objekty.

Vlastnosti jazyka:

- moderní objektově orientovaný jazyk,
- typová bezpečnost,
- automatická správa paměti – garbage collector,
- spolupráce s existujícími komponentami COM a DLL,
- podporuje zpracování chyb pomocí výjimek.

6.1.3 Microsoft Visual Studio 2010

Pro vývoj aplikace jsem zvolil vývojové prostředí Visual Studio 2010 Professional. K běhu Visual Studia je nutné mít nainstalován .NET Framework, který je součástí instalátoru.

Visual Studio usnadňuje nejen samotnou tvorbu aplikace, ale i její ladění. Při tvorbě okenních aplikací se používají formuláře, na které je možné pomocí tzv. Toolboxu přesouvat za pomoci myši různé komponenty. Toolbox je nabídka komponent seskupených do různých kategorií podle jejich vlastností. Umístěním komponenty na formulář se automaticky generuje příslušný zdrojový kód. Zdrojové kódy můžeme vytvářet a editovat prostřednictvím editoru kódu. Ten podporuje zvýrazňování syntaxe a automatické dokončování příkazů, což urychluje psaní kódu. Editor dále umožňuje otevření více zdrojových kódů a jejich zobrazení v jednom okně pomocí záložek.

K nalezení a odstranění chyb v programu slouží integrovaný debugger, který umožňuje program krokovat či jej zastavit v konkrétním místě.

6.1.4 Framework Mono

Pro zajištění přenositelnosti aplikace vyvíjené v rámci bakalářské práce jsem využil framework Mono [14]. Tento framework umožňuje vznik multiplatformních aplikací. Přenositelností aplikací se rozumí možnost jejich spuštění na různých operačních systémech a různých HW architekturách.

Framework Mono je projekt firmy Novell, jehož cílem je vytvořit sadu nástrojů kompatibilních s prostředím .NET. Jelikož se jedná o Open source software, můžeme se s Monem setkat v různých Linuxových distribucích jako je Ubuntu či Debian. Linux ovšem není jediný operační systém, pro který je framework Mono určen. Dalšími operačními systémy, které podporují framework Mono jsou Mac OS X, Solaris či Windows.

I přes neustálý vývoj frameworku Mono přetrvávají některé nedostatky. Při vytváření aplikací se setkáváme s drobnými odlišnostmi zobrazení v různých operačních systémech. Liší se vzhled a do jisté míry i chování Windows Forms. V aplikaci hojně využívaný dataGridView v některých případech nedokáže přizpůsobit velikost sloupců textu. V tomto případě musí uživatel pomocí myši a posuvníku změnit velikost těchto sloupců manuálně. Některé události z prostředí .NET nejsou

Monem podporovány vůbec. Úplný výpis známých chyb či nekompatibilit frameworku Mono je možné nalézt v tzv. bug listu na oficiálních stránkách výrobce [17].

Pro odhalení dosud neimplementovaných tříd, metod a dalších "sporných částí kódu", které není možné využívat v projektu Mono, slouží nástroj Mono Migration Analyzer (MoMA). Tento nástroj umožňuje analýzu aplikací psaných pro .NET Framework. Pomocí tohoto nástroje můžeme analyzovat soubory typu *.exe a *.dll.

Mono projekt disponuje dokonce vlastním vývojovým prostředím MonoDevelop. Toto prostředí lze oproti Visual Studiu použít při programování v různých operačních systémech (Linux, Mac OS, Windows či Solaris). MonoDevelop podporuje nejen programovací jazyk C#, ale i ostatní již výše zmíněné programovací jazyky určené pro .NET Framework.

6.2 Vlastní implementace

Pro implementaci podporovaných funkcí, které jsou definovány kapitolou [5] je klíčová komunikace mezi navrženou aplikací a RouterBOARDem. Tato komunikace je založena na principu: příkaz – provedení příkazu – odpověď. Aby byla tato komunikace možná, musí být nejprve stanoven komunikační protokol. Zařízení s RouterOS disponují komunikačním protokolem API.

6.2.1 API C# Třída

Základní API třída pro komunikaci s RouterOS napsána v programovacím jazyce C# byla převzata z webových stránek MikroTiku [7], kde je tato třída volně dostupná ke stažení. Tato třída se jmenuje MikrotikCLSS a obsahuje několik metod nezbytných pro přihlášení k RouterBOARDu, odeslání příkazů, příjmu odpovědí a ukončení komunikace. Jediným parametrem třídy MikrotikCLSS je řetězec obsahující IP adresu RouterBOARDu. Pro práci se síťovým protokolem TCP je použita třída TcpClient Frameworku .NET.

Pro přihlášení k RouterBOARDu slouží metoda *public bool Login(string username, string password)*. Parametry metody jsou řetězce představující uživatelské jméno a heslo.

Pro odeslání příkazu jsou zde dvě metody, které se liší pouze v odeslání slova nulové délky. Pokud příkaz obsahuje argumenty, používá se pro odeslání metoda *public void Send(string co)* pro odeslání příkazu a argumentů. Poslední argument se odesílá metodou *public void Send(string co, bool endsentence)*, která odešle argument a slovo nulové délky značící konec složeného příkazu.

Pro čtení odpovědí od RouterBOARDu slouží metoda *public List<string> Read()*.
Dále je použita třída *MD5CryptoServiceProvider* Frameworku .NET, která zabezpečí „zahešování" uživatelského hesla pomocí algoritmu MD5.

6.2.2 Odeslání příkazu

Vždy při otevírání nového okna se odesílá příkaz pro zjištění informací o funkci, která má být na daném formuláři zobrazena. Odpověď se okamžitě zpracuje a výsledkem je vyplněné okno. Například při otevření okna Firewall se odesílá příkaz */ip/firewall/filter/print*, odpovědí na tento příkaz je výpis všech pravidel firewallu.

Voláním metody *Send()* instance Mikrotik odešleme příkaz pro RouterBOARD.

Příklad odeslání příkazu:

```
Mikrotik.Send("/ip/dhcp-server/add");
Mikrotik.Send("=name=" + "server1");
Mikrotik.Send("=interface=" + "ether1");
Mikrotik.Send("=lease-time=" + "3d");
Mikrotik.Send("=address-pool=" + "pool1", true);
```

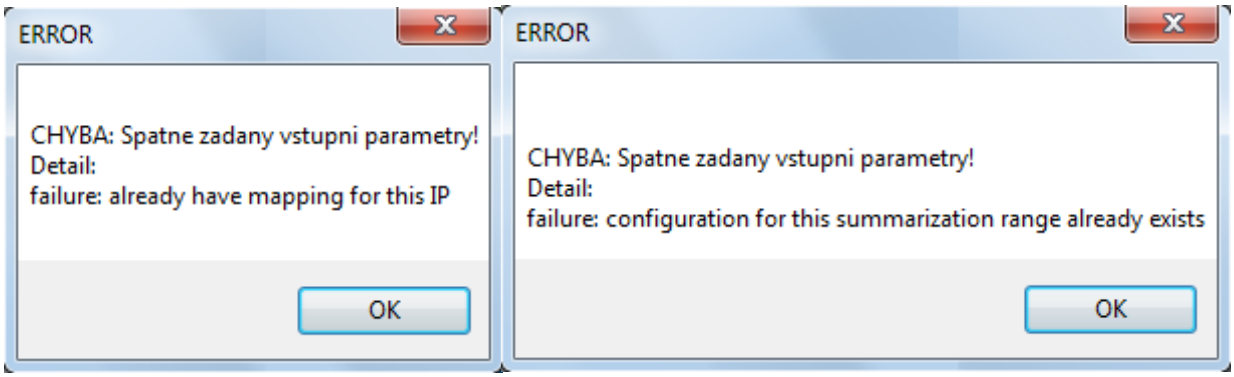
Tento sled příkazů slouží ke konfiguraci DHCP serveru. Z příkazů je patrné, že název tohoto serveru bude server1, rozhraní ether1 a expirační dobou výpůjčky IP adresy z rozsahu pool1 budou tři dny.

Po odeslání každého příkazu se kontroluje, zda RouterBOARD neodpověděl chybovou zprávou, k tomu slouží funkce *verify()*. Pokud RouterBOARD odpoví chybovou zprávou, uživateli se zobrazí dialogové okno (viz obrázek 6.2) s informací o chybě a přesným popisem této chyby v anglickém jazyce, kterým RouterBOARD odpověděl.

Ukázka zdrojového kódu funkce verify():

```
public string verify()
        {
            string s = "CHYBA: Spatne zadany vstupni
parametry!\nDetail:\n";

            string final = "";
            int pozice = 0;
            int ctrap = 0;
            foreach (string h in mikrotik.Read())
            {
                if (h.Contains("trap"))   //neuspech vykonani akce
                {
                    ctrap++;
                    pozice = h.IndexOf("message=");
                    final = h.Substring(pozice + 8); //ziskani tela zpravy
                    s += (final + '\n');
                }
            }
            if (ctrap > 0)
            {
                MessageBox.Show(s,"ERROR");  //zobrazeni chybove hlasky
                return "VERIFY_TRAP";
            }
            else
                return "VERIFY_OK";
        }
```



Obrázek 6.2: Ukázka chybových hlášení.

6.2.3 Zpracování odpovědí

Voláním metody *Read()* instance Mikrotik od RouterBOARDu získáme odpověď na příkaz. Odpověď je uložena v dynamickém datovém typu List. Jednotlivé prvky Listu jsou řetězce slov (věty). Aby bylo možné zpracovat odpověď, je nutné každou z těchto vět rozdělit na slova (parametry). Mezi slovy se nachází oddělovací znak „=“, díky kterému je možné použit funkci jazyka C# *Split()*, která rozdělí větu na slova. Slova jsou pak procházena jedno po druhém a testována zda se jedná o hledané slovo. Pokud se jedná o hledané slovo, víme, že následující slovo je hodnota parametru.

Příklad zpracování odpovědi:

```
foreach (string h in form1.Mikrotik.Read())
  {
      pom += h;
  }
  char delimiter = '=';
  string[] words = pom.Split(delimiter);        //rozdeleni vet na slova
  for (int i = 0; i < words.Count(); i++){      //prochazeni vsech slov
     //rozeznani slov a naplneni dataGridView
     if (words[i] == ".id")
     {
        dataGridView1.Rows.Add();
        dataGridView1.Rows[sloupec].Cells[0].Value = words[i + 1];
        sloupec++;
     }
     if (words[i] == "address")
     {
        dataGridView1.Rows[sloupec - 1].Cells[1].Value = words[i + 1];
     }
     if (words[i] == "disabled")
     {//cerveny krizek pro zakazane polozky a zelene zatrzitko pro aktivni
         if (words[i + 1] == "true")
            {
             dataGridView1.Rows[sloupec - 1].DefaultCellStyle.ForeColor =
               Color.Red;
             dataGridView1.Rows[sloupec - 1].Cells[5].Value =
               System.Drawing.Image.FromFile("cross.png");
            }
         else
            dataGridView1.Rows[sloupec - 1].Cells[5].Value =
             System.Drawing.Image.FromFile("check.png");
     }
  }
```

6.2.4 Editace

Editaci či otevření detailnějšího popisu zobrazeného nastavení je možné provádět dvojklikem na daný záznam. Tyto záznamy jsou zobrazeny pomocí komponent *dataGridView*, editovat lze pouze záznamy, které jsou k tomu určené, např. editace položek z výpisu historie provedených změn pochopitelně možná není. Pokud provedeme dvojklik nad editovatelným záznamem, vytvoří se již předvyplněné okno, aby bylo možné provést editaci záznamu snadno a rychle. Editace záznamu i otevření detailnějšího popisu je podmíněno správným rozeznáním záznamu, nad kterým se tyto operace provádí. Některé záznamy v sobě přímo obsahují název, například název DNS serveru je v celém výpisu jedinečný. Jiné záznamy ovšem jednoznačný název neobsahují, proto se při komunikaci s RouterBOARDdem musí použít jednoznačný identifikátor záznamu – id.

V následujících příkladech jsou ukázány požadavky na výpis konkrétních záznamů. Jako první je uveden přiklad příkazu pro výpis konkrétního rozhraní na kterém je spuštěn dynamický směrovací protokol OSPF. Jelikož každé rozhraní může být v rámci OSPF použito pouze jednou, je tento záznam možné identifikovat pomocí názvu rozhraní, který je uložen v proměnné *name*.

Příklad příkazu pro výpis OSPF:

```
Mikrotik.Send("/routing/ospf/interface/print");
Mikrotik.Send("?interface=" + name, true);
```

Při konfiguraci protokolu ARP můžeme jednomu rozhraní přiřadit několik různých IP adres, ale i stejnou IP adresu přiřadit více rozhraním. V takovém případě se pro identifikaci záznamu musí použít jednoznačný identifikátor záznamu.

Příklad pro výpis konkrétního ARP záznamu, identifikátor je uložen v proměnné id:

```
Mikrotik.Send("/ip/arp/print");
Mikrotik.Send("?.id=" + id, true);
```

Po přijetí výše zobrazených příkazů RouterBOARD odpoví vypsáním požadovaných informací. Tyto informace poté můžeme editovat. Po jejich editaci se odešle příkaz RouterBOARDu a ten provede změny. V posledním příkladu je ukázána editace DHCP serveru. Příkaz pro editaci záznamu je obdobný příkazu pro přidání záznamu, pouze místo příkazu *add* použijeme příkaz *set*.

Příklad editace názvu DHCP serveru:

```
Mikrotik.Send("/ip/dhcp-server/set");
Mikrotik.Send("=name=" + "server2, true");
```

Těmito příkazy jsme tedy přejmenovali název DHCP serveru na server2.

7 Ověření funkčnosti aplikace

Tato kapitola se zabývá testováním funkčnosti při vývoji jednotlivých funkcí aplikace. Přestože byly testovány všechny implementované funkce (viz kapitola 5.1), pro demonstraci funkčnosti aplikace jsem se rozhodl předvést pouze některé funkce. Pomocí snímků obrazovky zde bude ukázán postup konfigurace a výsledky budou konfrontovány s aplikací Winbox.

7.1 Použité RouterBOARDy

Pro testování funkčnosti aplikace jsem použil dva různé RouterBoardy RB 750G a RB 411AH.

- **RouterBOARD RB 750G** - jedná se o router malých rozměrů určený pro menší síť. Poskytuje pět ethernetových portů s rychlostí až 1000 Mbps. Má relativně výkonný procesor Atheros AR7161 (680 MHz) a paměť RAM 32 MB. Tento model obsahuje RouterOS v4.11. Jelikož nemá sériový port (RS-232), konfigurace se provádí pomocí ethernetu.
- **RouterBOARD RB 411AH** - tento model disponuje pouze jedním ethernetovým a jedním sériovým portem. Pro možnost připojení Wifi sítě je rozšířen o Wifi modul CM9. Procesor toho RouterBOARDu je stejný jako ve výše zmíněném modelu, paměť RAM je 64 MB. RouterOS tohoto modelu je verze 4.17. Konfiguraci lze provádět pomocí všech rozhraní, tedy ethernetu, Wifi i sériového portu.

7.2 Praktické ověření funkčnosti aplikace

Při tvorbě aplikace bylo nutné každou naimplementovanou funkci důkladně otestovat. Provedení veškerých změn konfigurace za použití mé aplikace jsem okamžitě ověřoval pomocí aplikace Winbox. Taktéž jsem ověřoval, jestli se všechny změny konfigurace provedené prostřednictvím aplikace Winbox projeví i v mé aplikaci.

Pro demonstraci funkčnosti aplikace jsem zvolil funkce určené v zadání bakalářské práce, tj. konfigurace DHCP, statický routing a DNS.

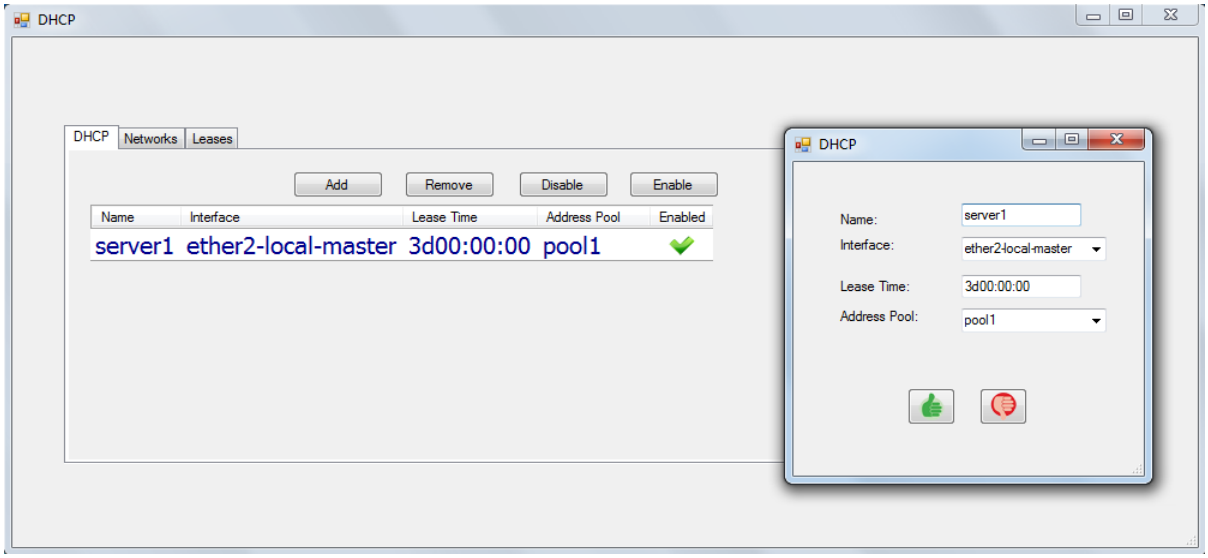
Ověřování funkčnosti aplikace jsem prováděl pod Operačními systémy MS Windows 7 a Ubuntu 10.04. Pro zjednodušení prezentace výsledků testování budu uvádět zobrazení konfigurace DHCP z prostředí MS Windows a zbylé funkce z prostředí Ubuntu. Ve všech případech jsem prováděl konfrontaci mojí aplikace s aplikací Winbox spuštěné v prostředí MS Windows.

Aplikaci je možné spustit ve všech operačních systémech s nainstalovaným frameworkem Mono. Těmito systémy může být Windows, Linux, Mac OS X nebo Solaris. Ke spuštění aplikace může posloužit např. příkazová řádka, kde stačí napsat „mono název_aplikace.exe“.

Verze operačního systému Ubuntu 10.04 obsahuje základní balíček Mono, přesto je nutné doinstalovat podporu okenních aplikací (WinForms). Toho je možné docílit použitím příkazové řádky nebo správce balíků Synaptic. Konkrétně se jedná o balík *libmono-winforms2.0-cil* a *libmono-accessibility2.0-cil*. Po doinstalování těchto balíků je možné aplikaci spustit.

7.2.1 Konfigurace DHCP serveru

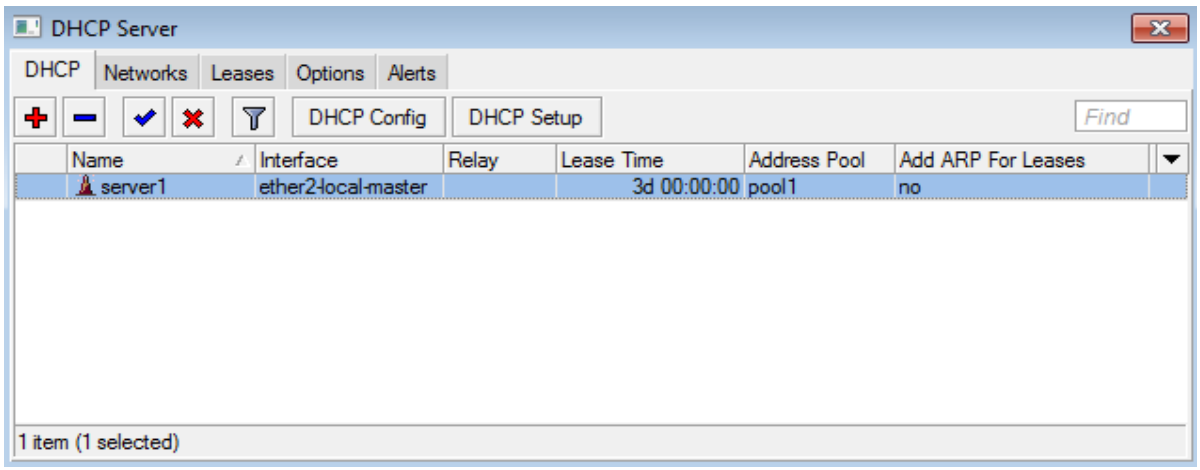
Před konfigurací DHCP serveru bylo nutné vytvořit IP Pool (menu IP – položka Pool), udávající rozsah IP adres z kterého bude DHCP server přidělovat adresy jednotlivým klientům. Na obrázku 7.1 je zobrazeno okno DHCP a nabídka ke konfiguraci parametrů DHCP serveru vyvolaná pomocí tlačítka Add.



Obrázek 7.1: Konfigurace DHCP serveru – MS Windows.

DHCP server jsem pojmenoval server1, expirační dobu zapůjčené IP adresy jsem zvolil tři dny a použil jsem adresy z rozsahu pool1. Takto nakonfigurovaný server jsem spustil na rozhraní ether2-local-master.

Pokud požadujeme přidělování adresy statické místo dynamické, zvolíme v nabídce Address Pool možnost static-only. Poté by server DHCP přiděloval IP adresy definované ve třetí záložce – Leases. V záložce Networks lze nastavit údaje přidělované DHCP serverem jako je výchozí brána nebo DNS Server. Konfiguraci jsem prováděl z operačního systému MS Windows

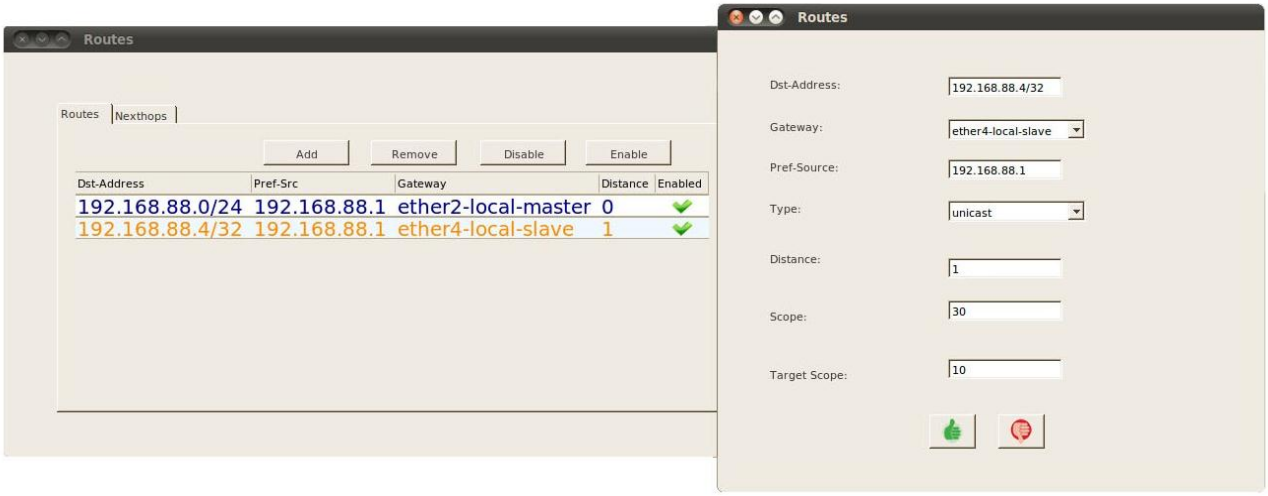


Obrázek 7.2: Ověření nastavení DHCP serveru – Winbox.

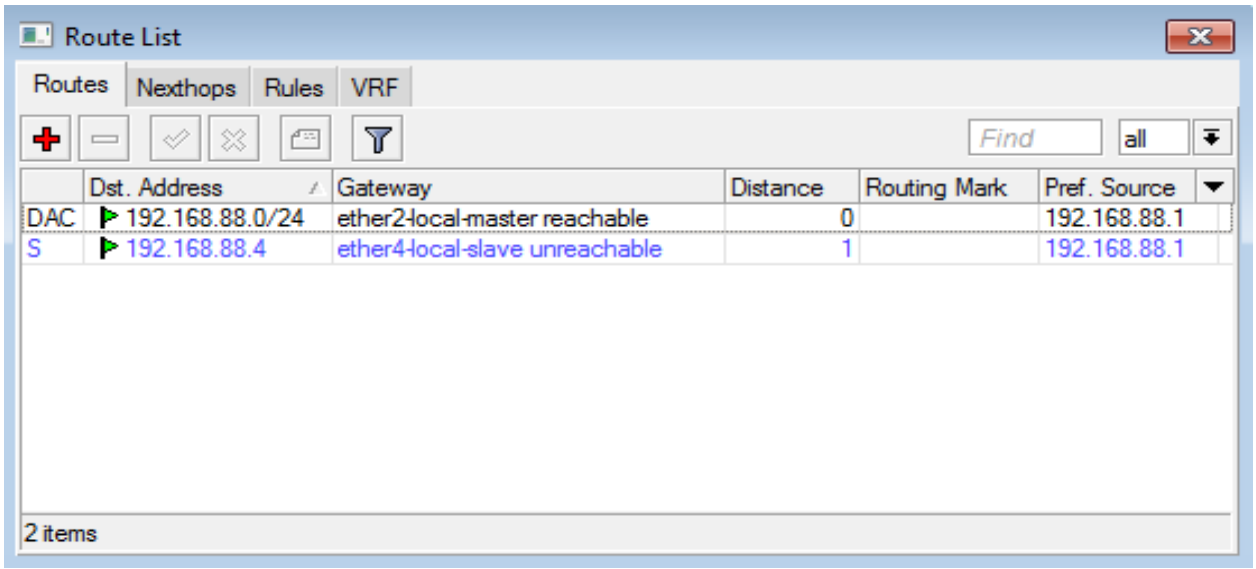
Na obrázku 7.1 jsou zobrazeny okna pro konfiguraci DHCP serveru z prostředí MS Windows. Na obrázku 7.2 je ukázáno, že se provedené změny nastavení projevily i v aplikaci Winbox.

7.2.2 Statický routing

Při vytváření statické cesty sítí jsem zadal cílovou IP adresu sítě 192.168.88.4 a výchozí bránu ether4-local-slave. Jak můžeme vidět na obrázku 7.3, tato cesta se zbarvila oranžově, což značí nedosažitelnou síť. K ether4-local-slave totiž fyzicky není připojeno žádné zařízení. V záložce Nexthops se zobrazují adresy sousedních routerů. Konfiguraci jsem prováděl z operačního systému Ubuntu 10.04.



Obrázek 7.3: Konfigurace statického směrování – Ubuntu.

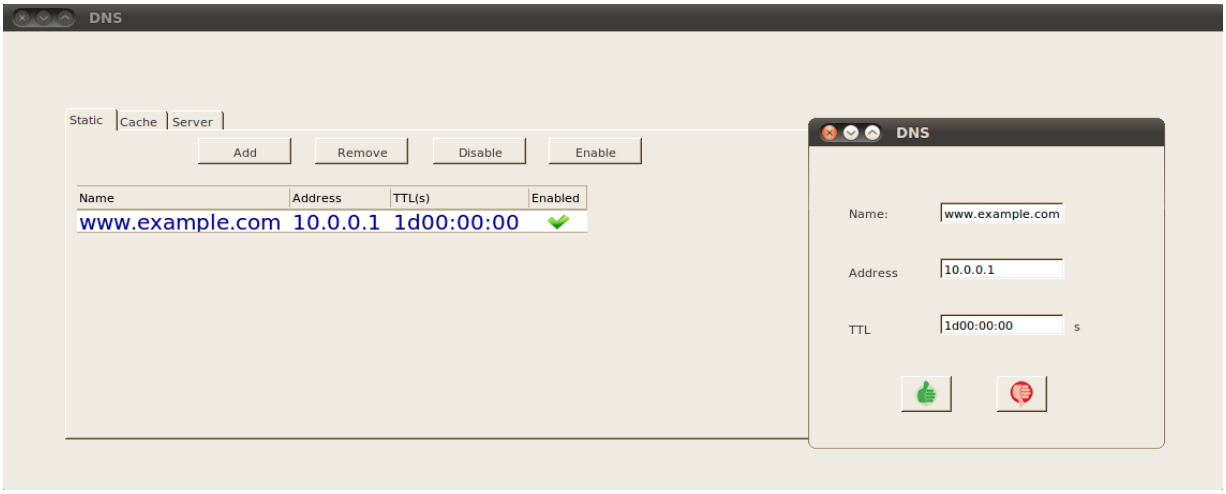


Obrázek 7.4: Ověření nastavení statického směrování – Winbox.

Na obrázku 7.3 jsou zobrazeny okna pro konfiguraci statického směrování z prostředí Ubuntu. Na obrázku 7.4 je ukázáno, že se provedené změny projevily i v aplikaci Winbox.

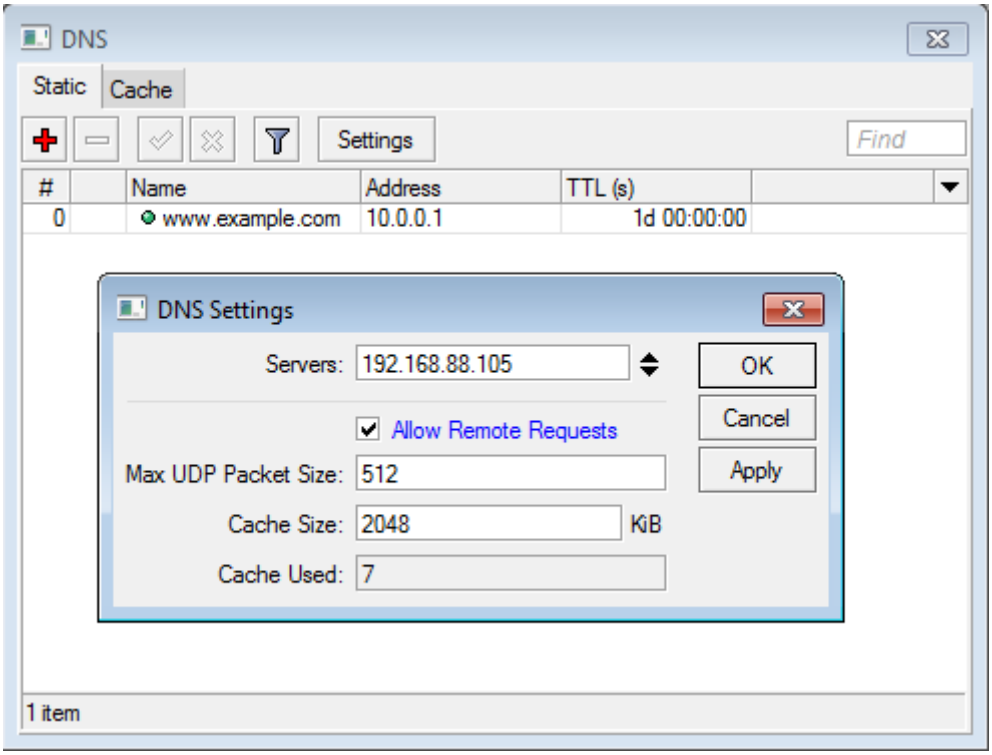
7.2.3 DNS Server

Na obrázku 7.5 je zobrazeno vytvoření statického DNS záznamu. Pomocí tlačítka Add jsem vyvoval nabídku pro vytvoření statického záznamu. Pro www.example.com jsem přiřadil IP adresu 10.0.0.1. Dobu životnosti tohoto statického záznamu jsem zvolil jeden den. V Záložce Server jsem provedl nastavení DNS Serveru, které je zobrazené na obrázku 7.6. Záložka Cache slouží k zobrazení záznamů vyrovnávací paměti, pomocí kterých se zrychluje činnost pří vyřizování požadavků.



Obrázek 7.5: Konfigurace statického záznamu DNS – Ubuntu.

Obrázek 7.6: Konfigurace DNS serveru – Ubuntu.



Obrázek 7.7: Ověření nastavení DNS serveru – Winbox.

Na obrázcích 7.5 a 7.6 jsou zobrazena okna pro konfiguraci DNS serveru z prostředí Ubuntu. Na obrázku 7.7 je ukázáno, že se provedené změny projevily i v aplikaci Winbox.

Testování funkčnosti prokázalo, že vytvořená aplikace komunikuje správně s RouterBOARDem a provádí požadovaná nastavení konfigurace. Ve všech testovaných případech byla konfigurace RouterBOARDu detekována shodně v obou aplikacích. Testování také prověřilo činnost aplikace pod různými operačními systémy.

8 Závěr

V této kapitole se nachází zhodnocení splnění vytyčených cílů mojí práce a nástin možných rozšíření vytvořené aplikace. Cílem práce bylo vytvořit aplikaci pro vzdálenou správu zařízení se systémem MikroTik RouterOS. Pro splnění cílů bylo nutné se seznámit se zařízeními firmy MikroTik zvanými RouterBOARDy a s možnostmi jejich správy. Často využívaným prostředkem pro vzdálenou správu těchto zařízení je aplikace Winbox vytvořená firmou MikroTik. Tato aplikace nabízí možnost plné správy RouterBOARDu. Je však platformově závislá na MS Windows. Proto byl při tvorbě vlastní aplikace kladen důraz na použitelnost aplikace při konfiguraci základních i pokročilých funkcí RouterOS, nejen pod operačním systémem MS Windows, ale i Linux.

Před samotným návrhem aplikace jsem musel nastudovat vlastnosti aplikace Winbox pro získání přehledu a pochopení poskytovaných funkcí. Důležité bylo také důkladné prostudování protokolu API. Poté jsem vytvořil grafický návrh aplikace a postupně jsem implementoval zvolné funkce aplikace.

Aplikaci jsem vytvářel v prostředí MS Windows pomocí vývojového prostředí Visual Studio. Pro přenositelnost aplikace mezi operačními systémy MS Windows a Linux jsem použil framework Mono. Výsledkem je platformově nezávislá aplikace.

Aby bylo možné ověřit funkčnost aplikace, bylo nezbytné otestovat všechny implementované funkce. Ověřování funkčnosti aplikace probíhalo pod operačními systémy MS Windows a Linux za použití dvou různých RouterBOARDů (viz kapitola 7).

Přínos mojí práce vidím ve vzniku nové, platformově nezávislé aplikace, která umožňuje nastavení potřebných funkcí RoterBOARDu a jiných zařízení s RouterOS. Tato aplikace se tak stává alternativou aplikace Winbox pro správu počítačových sítí malých firem a domácností založených na síťových prvcích MikroTik, které získávají stále větší popularitu mezi uživateli. Spolu s teoretickým rozborem poskytovaných funkcí tvoří užitečný nástroj pro zdatnější uživatele PC, kteří si chtějí konfigurovat svou počítačovou síť sami.

8.1 Budoucnost projektu

Přestože cíle kladené na aplikaci byly splněny, stále je zde prostor pro vylepšování a přidávání nových funkcionalit aplikace. Nyní aplikace podporuje připojení se k RouterBOARDu pouze pomocí IP adresy zařízení, proto by mohla být přidána možnost připojit se pomocí MAC adresy zařízení. Jelikož při výpadku napájení dojde k vynulování systémového času, další užitečnou funkcí by mohl být NTP klient pro synchronizaci hodin zařízení dle zvoleného NTP serveru. Logovací záznamy by tak byly zaznamenávány se správnou časovou značkou i po výpadku napájení. Aplikace podporuje vytváření zálohy nakonfigurovaného systému. Tato záloha se uloží do paměti RouterBOARDu, rozšířením by mohla být možnost uložení vytvořené zálohy do PC pomocí protokolu ftp či tftp.

Literatura

[1] PUŽMANOVÁ, Rita. *Moderní komunikační sítě od A do Z*. Praha: Computer Press, 1998, 446 s. ISBN 80-722-6098-7.

[2] KABELOVÁ, Alena a Libor DOSTÁLEK. *Velký průvodce protokoly TCP/IP a systémem DNS*. 5., aktualiz. vyd. Brno: Computer Press, 2008, 488 s. ISBN 978-80-251-2236-5.

[3] KAČMÁŘ, Dalibor. *Programujeme*: *NET aplikace ve Visual Studiu .NET*. Praha: Computer Press, 2001, 335 s. ISBN 80-722-6569-5.

[4] HANÁK, Ján. *Praktické objektové programování v jazyce C# 4.0*. Vyd. 1. Brno: Artax, 2009, 180 s. Microsoft (Artax). ISBN 978-80-87017-07-4.

[5] MikroTik. *MikroTik Routers and Wireless* [online]. 1997 [cit. 2012-04-11]. Dostupné z: http://www.mikrotik.com/index.html

[6] MikroTik Wiki. *Manual:API* [online]. 2008, modified on 14 February 2012 [cit. 2012-04-11]. Dostupné z: http://wiki.mikrotik.com/wiki/Manual:API

[7] MikroTik Wiki. *API in C Sharp* [online]. 2008, modified on 14 July 2009 [cit. 2012-04-11]. Dostupné z: http://wiki.mikrotik.com/wiki/API_in_C_Sharp

[8] MikroTik Wiki. *Manual:License* [online]. 2008, modified on 20 January 2012 [cit. 2012-04-11]. Dostupné z: http://wiki.mikrotik.com/wiki/Manual:License

[9] MikroTik Wiki. *Manual:TOC* [online]. 2008, modified on 4 January 2012 [cit. 2012-04-11]. Dostupné z: http://wiki.mikrotik.com/wiki/Manual:TOC

[10] MikroTik. *WiFi router přímo od výrobce* [online]. 2007 [cit. 2012-04-11]. Dostupné z: http://mikrotik.cz/

[11] Orisoft Ltd. *.NET* [online]. [cit. 2012-04-11]. Dostupné z: http://www.orisoft.co.uk/Home/DotNet

[12] Lupa. *Bráníme se odposlechu: obrana na switchi* [online]. 2006 [cit. 2012-04-11]. Dostupné z: http://www.lupa.cz/clanky/branime-se-odposlechu-obrana-na-switchi/

[13] Windows IT Pro. *What's Network Address Translation (NAT)?* [online]. 2003 [cit. 2012-04-11]. Dostupné z: http://www.windowsitpro.com/article/tcpip/what-s-network-address-translation-nat-

[14] Mono-project. *Mono* [online]. 2004 [cit. 2012-04-11]. Dostupné z: http://www.mono-project.com/Main_Page

[15] RouterBoard. *RouterBoard* [online]. 2002 [cit. 2012-04-11]. Dostupné z: http://routerboard.com/?group id=11

[16] CCNPGUIDE. *CCNP ROUTE 642-902 :: OSPF* [online]. 7.2.2011 [cit. 2012-04-11]. Dostupné z: http://www.ccnpguide.com/ccnp-route-642-902-ospf/

[17] Mono-project. *Bugs* [online]. 2004 [cit. 2012-04-18]. Dostupné z: http://www.mono-project.com/Bugs

Seznam příloh

Příloha A

Obsah CD

K bakalářské práci je přiloženo CD obsahující:

- Solution – adresář zdrojových souborů programu vytvořených ve Visual Studiu 2010.
- Docs – adresář s programovou dokumentaci vygenerovanou pomocí nástroje Doxygen.
- BP – technická zpráva ve formátu pdf.
- BP – technická zpráva ve formátu doc.
- Manual – pokyny k použití aplikace.

Příloha B

Metriky kódu

Počet souborů: 153

Počet řádků kódu: 11249

Velikost zdrojového textu: 1537 kB

Velikost binárního souboru: 566 kB