VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ
ÚSTAV TELEKOMUNIKACÍ

FACULTY OF ELECTRICAL ENGINEERING AND COMMUNICATION
DEPARTMENT OF TELECOMMUNICATIONS

NÍZKOÚROVŇOVÉ ŘÍZENÍ A SBĚR DAT V BEZDRÁTOVÉM PŘÍSTUPOVÉM BODU MIKROTIK

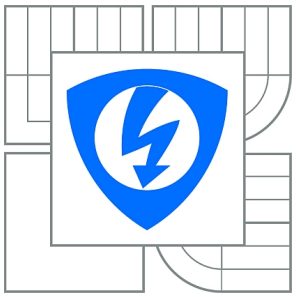
BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE MICHAL JURČÍK
AUTHOR

BRNO 2011

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ

ÚSTAV TELEKOMUNIKACÍ

FACULTY OF ELECTRICAL ENGINEERING AND COMMUNICATION

DEPARTMENT OF TELECOMMUNICATIONS

NÍZKOÚROVŇOVÉ ŘÍZENÍ A SBĚR DAT V BEZDRÁTOVÉM PŘÍSTUPOVÉM BODU MIKROTIK

LOW LEVEL CONTROL AND DATA ACQUISITION IN THE WIRELESS ACCESS POINT MIKROTIK

BAKALÁŘSKÁ PRÁCE

BACHELOR'S THESIS

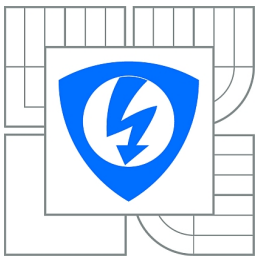
AUTOR PRÁCE — MICHAL JURČÍK

AUTHOR

VEDOUCÍ PRÁCE — Ing. LUKÁŠ RŮČKA

SUPERVISOR

BRNO 2011



VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

Fakulta elektrotechniky a komunikačních technologií

Ústav telekomunikací

Bakalářská práce

bakalářský studijní obor
Teleinformatika

Student:	Michal Jurčík	***ID:***	106511
Ročník:	3	***Akademický rok:***	2010/2011

NÁZEV TÉMATU:

Nízkoúrovňové řízení a sběr dat v bezdrátovém přístupovém bodu MikroTik

POKYNY PRO VYPRACOVÁNÍ:

Seznamte se s přístupovým bodem MikroTik RB433, jeho operačním systémem RouterOS a možnostmi konfigurace a využití protokolu IPv6 na tomto zařízení. Nalezněte a popište možnosti přístupu k interním funkcím a komponentám přístupového bodu. Popište možnosti řízení vlastností přístupového bodu pomocí interních funkcí. Zaměřte se především na možnost získávání informací o spojeních mezi přístupovým bodem a mobilními stanicemi. Nalezněte vhodný způsob získávání těchto informací a jejich zpracování v přístupovém bodě. Dále navrhněte způsob propagace těchto informací v reálném čase na ostatní přístupové body pomocí bezdrátového spojení. Podrobně popište a zdokumentujte zjištěné informace a použité postupy.

DOPORUČENÁ LITERATURA:

[1] Hallian, C.: Embedded Linux Primer: A Practical Real-World Approach. Indiana: Prentice Hall, 2006. 576 s. ISBN: 0-13-167984-8.
[2] Harbison, S., Steele, G.: Referenční příručka jazyka C. Veletiny: Science, 1996. 334 s. ISBN: 80-901475-50.
[3] MikroTik Wiki [online]. 2010 [cit. 2011-02-02]. Dostupný z WWW: <http://wiki.mikrotik.com/wiki/Main_Page>.

Termín zadání:	7.2.2011	***Termín odevzdání:***	2.6.2011

Vedoucí práce: Ing. Lukáš Růčka

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

ABSTRAKT

Bakalářská práce je zaměřena na využití protokolu ICMP pro přenos uživatelsky definovaného obsahu dat. Hlavním cílem bakalářské práce je seznámení s využitím socketů operačního systému OpenWRT, pomocí nichž jsou přenášeny zprávy protokolu ICMP. Bakalářská práce je rozdělena do dvou částí. První část obsahuje popis funkcí, maker a nastavení vytvořeného programu. V druhé části pak popis komunikace mezi přístupovými body.

KLÍČOVÁ SLOVA

přístupový bod, operační systém, skriptovací jazyk, konzole, socket, WDS, protokol ICMP

ABSTRACT

Bachelor thesis is focused on using the ICMP protocol for transferring user-defined data content. The main objective of this thesis is to introduce of using sockets working with OpenWRT operating system, which transmitting ICMP messages. Bachelor thesis is divided into two parts. The first section contains a description of program functions, macros and settings. In the second part there is a description of communication between access points.

KEYWORDS

access point, operation system, scripting language, console, socket, WDS, protocol ICMP

MICHAL JURČÍK, *Nízkoúrovňové řízení a sběr dat v bezdrátovém přístupovém bodu MikroTik*: bakalářská práce. Brno: Vysoké učení technické v Brně, Fakulta elektrotechniky a komunikačních technologií, Ústav telekomunikací, 2011. 62 s. Vedoucí práce byl Ing. Lukáš Růčka

PROHLÁŠENÍ

Prohlašuji, že svou bakalářskou práci na téma „Nízkoúrovňové řízení a sběr dat v bezdrátovém přístupovém bodu MikroTik“ jsem vypracoval samostatně pod vedením vedoucího bakalářské práce a s použitím odborné literatury a dalších informačních zdrojů, které jsou všechny citovány v práci a uvedeny v seznamu literatury na konci práce.

Jako autor uvedené bakalářské práce dále prohlašuji, že v souvislosti s vytvořením této bakalářské práce jsem neporušil autorská práva třetích osob, zejména jsem nezasáhl nedovoleným způsobem do cizích autorských práv osobnostních a jsem si plně vědom následků porušení ustanovení § 11 a následujících autorského zákona č. 121/2000 Sb., včetně možných trestněprávních důsledků vyplývajících z ustanovení § 152 trestního zákona č. 140/1961 Sb.

Brno ....

(podpis autora)

OBSAH

SEZNAM OBRÁZKŮ

SEZNAM TABULEK

ÚVOD

Přístupový bod k bezdrátové síti je zařízení, ke kterému se jednotliví klienti připojují. Klienti spolu nekomunikují přímo, ale prostřednictvím přístupového bodu, takže mohou být jednodušší z hlediska řízení a nemusí být ve vzájemném rádiovém spojení. Přístupové body dále mohou upřesňovat a specifikovat nastavení pro jednotlivé klienty, ať se již jedná o možnost připojení k přístupovému bodu, omezení datového toku či změny priority přenosu.

V práci je použito přenosu zpráv pomocí protokolu ICMP. ICMP protokol, definovaný v RFC 792, je součást sady protokolů internetu. ICMP zprávy se typicky generují při chybách v IP datagramech (specifikováno v RFC 1122) nebo pro diagnostické a směrovací účely. Protokol obecně neslouží pro přenos dat, jelikož nepatří v TCP/IP modelu sítí do transportní vrstvy, nýbrž do vrstvy IP.

Pro praktickou ukázku je znázorněn vývoj programu pro přenos ICMP zpráv společně s uživatelsky definovaným obsahem. Tyto zprávy obsahují informace o parametrech přístupových bodů, které mohou být dále zpracovány a použity například pro změnu jejich nastavení.

1 ZAŘÍZENÍ

1.1 Výběr přístupového bodu

V dnešní době je možné vybírat ze široké škály zařízení, která jsou pro účely sítě vhodná. Pro nízkoúrovňové řízení přístupového bodu, které má zpracovávat, ukládat a interpretovat získaná data, je nutné si uvědomit, že náročnost těchto procesů z hlediska možnosti snížení výkonu daného přístupového bodu, může mít za následek kolísání přenosové rychlosti, zvýšení reakční doby nebo vliv na stabilitu. Proto je nutné uvažovat takové zařízení, jehož výkon co nejvíce vyhovuje požadavkům. Mezi vhodnými zařízeními figurují výrobci Asus, Ovislink, TP-Link, Linksys a zejména výrobci D-Link a MikroTik, protože pro zařízení těchto výrobců již existují funkční operační systémy založené na volně šiřitelných Linuxových distribucích. Pro tyto účely je vybrán přístupový bod firmy MikroTik, konkrétně model RB433 s bezdrátovou kartou R52Hn pracující se standardy IEEE 802.11a, 802.11b, 802.11g a 802.11n.

1.1.1 Standard IEEE 802.11

IEEE 802.11 je Wi-Fi (*Wireless Fidelity*) standard s dalšími doplňky pro lokální bezdrátové sítě (*Wireless LAN*, *WLAN*). Výraz 802.11x je používán pro množinu doplňků k tomuto standardu. Výraz IEEE 802.11 je také spojován s původním 802.11 standardem, tedy bez dalších doplňků.

Standard 802.11 zahrnuje šest druhů modulací pro vysílání radiového signálu, přičemž všechny používají stejný protokol. Nejpoužívanější modulace jsou definované v dodatcích k původnímu standardu s písmeny *a*, *b* a *g*. Doplněk 802.11n přináší další techniku modulace, která studuje různé možnosti nastavení parametrů fyzické vrstvy a MAC (*Medium Access Control*) podvrstvy pro zvýšení datové propustnosti. Mezi tyto možnosti patří použití více antén, změny kódovacích schémat a změny MAC protokolů. Aktuální cíl skupiny je přenosová rychlost minimálně 100 Mbit/s nad MAC vrstvou. Navíc má IEEE 802.11n zajistit vyšší dosah se zachováním co největší rychlosti a zvýšit odolnost proti rušení.

Původní zabezpečení standardu 802.11 bylo vylepšeno dodatkem *i*. Další dodatky (*c–f*, *h*, *j*) pouze opravují nebo rozšiřují předchozí specifikaci.

Standardy 802.11b a 802.11g používají frekvenční pásmo 2,4 GHz. Proto mohou zařízení interferovat s mikrovlnnými troubami, bezdrátovými telefony, s Bluetooth nebo s dalšími zařízeními používajícími stejné pásmo. Oproti tomu standard 802.11a používá 5 GHz pásmo a není tedy ovlivněn zařízeními pracujícími v pásmu 2.4 GHz.

Přehled jednotlivých standardů 802.11x

802.11a

Tento standard využívá Wi-Fi v pásmu 5 GHz. Používá modulaci OFDM (*Orthogonal Frequency Division Multiplexing*). Oproti standardu IEEE 802.11b/g je tento stabilnější a vyspělejší. Má větší povolený vyzařovací výkon oproti 802.11b/g, tím ho lze používat na delší vzdálenosti.

802.11b

Tento standard je jedním z doplňků norem IEEE 802.11 zabývajících se definicí bezdrátového komunikačního standardu známým pod komerčním názvem Wi-Fi. Byl schválen v roce 1999 a oproti původnímu standardu navyšuje přenosovou rychlost na 11 Mbit/s v přenosovém pásmu 2,4 GHz.

802.11c

IEEE 802.11c je Wi-Fi standard věnující se přemosťování v bezdrátových zařízeních. Jde o hotový standard doplňující standard IEEE 802.1D, který přidává požadavky na přemosťování MAC vrstvy, což je podvrstva linkové vrstvy. Standard IEEE 802.1D upravuje základní LAN standard pro 802.11 rámce. Zejména dodává do klauzule 2.5 SISS (*Support of the Internal Sublayer Service*) podklauzuli, která pokrývá přemosťovací operace v rámci 802.11 MAC podvrstvy.

802.11d

IEEE 802.11d je Wi-Fi standard často nazývaný také jako globální harmonizační standard. Je používaný v zemích, kde nejsou povoleny systémy používající jiné dodatky k IEEE 802.11 standardu.

Definuje požadavky na fyzickou vrstvu k uspokojení regulačních domén nepokrytých existujícími standardy. Liší se v povolených frekvencích, vyzařovacích výkonech a propustnosti signálu. Specifikace eliminuje nutnost vývoje a výroby specifických produktů pro různé země.

Chování protokolu

Zapnutím podpory pro IEEE 802.11d v přístupovém bodě způsobí, že zařízení začne vysílat do celé sítě ISO (*International Organization for Standardization*) kód země, ve které se nachází, jako součást svých beacon paketů a požadavků na odpověď. Pokud je zapnut, klient přizpůsobí své frekvence, vyzařovací výkon a propustnost. Standard je tak vhodný pro systémy, které chtějí poskytovat globální roaming.

802.11e

IEEE 802.11e je Wi-Fi doplněk standardu IEEE 802.11 vylepšující takzvanou MAC podvrstvu linkové vrstvy rozšířením podpory kvalitu služeb QoS (*Quality of Service*). Standard je důležitý pro aplikace citlivé na zpoždění jako jsou VoWIP (*Voice over Wireless IP*) a proudové multimédia.

802.11g

Je Wi-Fi standard rozšiřující IEEE 802.11b. Je zpětně kompatibilní, vysílá ve stejném frekvenčním pásmu 2400 — 2485 MHz, ale maximální nominální rychlost je 54 Mbit/s, což odpovídá přenosům přibližně o rychlosti 25 Mbit/s.

Použité modulační schéma je OFDM pro přenosové rychlosti 6, 9, 12, 18, 24, 36, 48 a 54 Mbit/s, přičemž pro rychlosti 1, 2, 5.5 a 11 Mbit/s je použito stejné schéma jako ve standardu IEEE 802.11b. Vysílací výkon je snížen oproti IEEE 802.11b z 200 mW na 65 mW.

802.11h

IEEE 802.11h je Wi-Fi standard doplňující standard 802.11a, který je navržen s ohledem na evropské podmínky, aby bylo možné sítě využívat mimo budovy. Řeší například problémy s rušením od ostatních zařízeních pracujících na 5 GHz frekvenci. Na tomto pásmu pracují například radary nebo některé satelitní systémy. V podstatě mají bezdrátová zařízení v případě, že detekovaly rušení, omezit vysílací výkon nebo uvolnit kanál, na kterém toto rušení rozpoznaly.

Tento standard upravuje fyzickou vrstvu a podčást linkové vrstvy, takzvanou MAC podvrstvu. Dynamickým výběrem kanálu přináší také lepší pokrytí jednotlivých kanálů.

802.11n

IEEE 802.11n je Wi-Fi standard, který si klade za cíl upravit fyzickou vrstvu a podčást linkové vrstvy, takzvanou MAC podvrstvu tak, aby se docílilo reálných rychlostí přes 100 Mbit/s. Nicméně maximální rychlost může být až 600 Mbit/s. Měl by se také zvýšit dosah.

Zvýšení rychlosti se dosahuje použitím MIMO (*Multiple Input Multiple Output*) technologie, která využívá vícero vysílacích a přijímacích antén. Citace dle [1].

Tab. 1.1: Přehled standardů IEEE 802.11

Standard	**Rok vydání**	**Pásmo** [GHz]	**Maximální rychlost** [Mbit/s]
IEEE 802.11	1997	2,4	2
IEEE 802.11a	1999	5	54
IEEE 802.11b	1999	2,4	11
IEEE 802.11g	2003	2,4	54
IEEE 802.11y	2008	3,7	54
IEEE 802.11n	2009	2,4 nebo 5	600

1.1.2 Technické parametry

Vybraný přístroj obsahuje mikroprocesor Atheros s frekvencí jádra 300 MHz založený na architektuře zkrácené instrukční sady MIPS-BE (*Microprocessor without Interlocked Pipeline Stages*), operační pamětí 64 MB RAM a vnitřní pamětí rovněž 64 MB RAM. Dále obsahuje 3 LAN porty a 3 miniPCI sloty, které jsou důležité z důvodu rozšíření vnitřní paměti, ale především pro instalaci bezdrátové karty, bez níž nelze přístroj použít jako bezdrátový přístupový bod. Je použita bezdrátová karta R52Hn podporující standardy 802.11a/b/g/n s přenosovou rychlostí až 300 Mbit/s a propustností 200 Mbit/s na uživatele v obou směrech toku dat.

1.1.3 Operační systém

Přístupový bod MikroTik RB433 obsahuje již předinstalovaný operační systém RouterOS verze 4.10. Oproti svým předchůdcům z řady 3.x obsahuje mnohá vylepšení a také změny týkající se předinstalovaných balíčků a možnostech využití rozhraní API (*Application Programming Interface*) pro programování vnitřních aplikací operačního systému.

Jinou možností je použití volně šiřitelných distribucí operačních systémů založených zejména na Linuxovém jádře. Z výčtu všech dostupných systémů je nejlepší volbou použití OpenWRT distribuce, která je dostupná i na webových stránkách výrobce. Tyto systémy dovolují plné řízení přístupového bodu a rovněž implementaci vlastních programů na úrovni vyšších programovacích jazyků. Jde o velkou výhodu, protože zkušení uživatelé mohou prakticky z daného přístroje vytvořit menší počítač. Nevýhodou těchto systémů však může být uživatelova neznalost, která snadno vyústí k selhání směrovače nebo v extrémních případech k jeho zničení.

V práci je použit stávající RouterOS a pro manipulaci s daty implementovaný skriptovací jazyk, pracující s interními funkcemi systému. Za zmínku ještě stojí

možnost využití funkce metarouter, kterou RouterOS nabízí. Ta umožňuje vytvoření virtuálního systému běžícího současně se systémem statickým. Takto vytvořený systém může být právě OpenWRT, doplňující samotný RouterOS o možnosti, které nenabízí nebo s kterými si neumí poradit.

2 SKRIPTOVACÍ JAZYK

2.1 Popis

RouterOS obsahuje již vestavěný výkonný skriptovací jazyk. Skripty poskytují administrátorům způsob, jak automatizovat údržbu systému spojenou s nastávajícími událostmi. Skripty mohou být ukládány do skriptovacích repositářů nebo psány přímo do systémové konzole. Mezi události, spouštějící samotné skripty, patří zejména systémový plánovač úloh, umožňující spouštění zápisů v přesně daných časových intervalech.

2.2 Struktura

RouterOS skript je rozdělen na počet příkazových řádků. Jednotlivé řádky příkazů jsou vykonávány jeden po druhém, dokud není dosaženo konce samotného skriptu nebo nenastane běhová chyba.

2.2.1 Příkazový řádek

Systémová konzole používá následující příkazovou konstrukci:

Tab. 2.1: Konzolová syntaxe

[**prefix**] [**path**] **command** [**uparam**] [**param1=**[**value**]] .. [**paramN=**[**value**]]	
[**prefix**]	znak „:“ nebo „/“ rozlišující, zda se jedná o příkaz nebo relativní cestu
[**path**]	relativní cesta k příslušnému menu konzole
command	jeden z příkazů dostupný v příslušném menu
[**uparam**]	nepovinný parametr, ačkoli musí být použit, pokud je vyžadován příkazem
[**param**]	konkrétní parametry následované příslušnými hodnotami
[**value**]	hodnota parametru

Konec každého řádku je zastoupen znakem „;“ nebo *NEWLINE*, které však nejsou vždy vyžadovány. Samotný příkaz uvnitř (), [] nebo {} závorek, nevyžaduje ukončovací znak, jelikož závorky charakterizují jeho začátek a konec. Každý příkaz uvnitř dalšího příkazu začíná a končí hranatými závorkami. Navíc lze příkazový řádek vytvořit z více fyzických řádků pomocí daných pravidel viz níže.

2.2.2 Fyzický řádek

Fyzický řádek je sekvence znaků ukončených sekvencí EOL (*End of Line*). Lze použít jakoukoli ukončovací sekvenci standardních platforem Unix, Windows a Mac nebo použít znaku „\n“, jenž je standardní konverzí jazyka C mezi jednotlivými základnami.

2.2.3 Komentáře

Komentáře začínají dvojitým křížkem „#“ a končí koncem fyzického řádku. Mezery a jiné symboly před znakem „#“ nejsou akceptovány. Komentáře jsou syntaxí ignorovány. Jestliže se dvojitý křížek objeví uvnitř řetězce znaků, není považován za komentář.

2.2.4 Spojování řádků

Dva a více fyzických řádků může být spojeno v jeden řádek logický použitím znaku „\“. Řádky končící tímto lomítkem však nemohou rozvíjet komentáře a rovněž nemohou stát kdekoliv jinde na řádku, než uvnitř řetězce znaků.

2.2.5 Symboly

Mezi symboly patří všechna klíčová slova, proměnné a znaky použity v jazyce RouterOS. Každý z těchto symbolů může být oddělen mezerami. Výjimkou je případ, kdy dojde ke sloučení symbolů a výsledný výraz je interpretován jako jiný nebo neznámý symbol. V tomto případě je nutností použít mezery jako oddělovače. Tu však nemůžeme vložit mezi klíčové slovo a znak „=“.

2.2.6 Pole působnosti

Proměnné mohou být použity pouze v určitých oblastech skriptu. Jedná se o oblasti působnosti, které určují viditelnost dané proměnné. Existují oblasti dvě a to globální a lokální. Proměnná deklarovaná v určitém bloku skriptu je přístupná pouze v tomto bloku od její deklarace po ukončení bloku. Se skončením bloku proměnná zaniká.

Globální neboli kořenová oblast je základní oblastí skriptu. Je vytvořena automaticky a nelze ji zrušit.

Lokální oblast je soubor uživatelem definovaných bloků z nichž každý je uzavřen ve složených závorkách. Vytvořením globální proměnné uvnitř lokální oblasti způsobí rovněž její zrušení po skončení činnosti daného bloku.

2.3 Datové typy

Skriptovací jazyk RouterOS využívá tyto datové typy proměnných:

Tab. 2.2: Datové typy skriptovacího jazyka

Typ	Význam
number	64 bitové znaménkové celé číslo (signed Integer)
boolean	hodnoty pravda – nepravda (true – false)
string	řetězec znaků
internal ID	hexadecimální hodnota s prefixem „*“, která identifikuje jednotlivé položky v konzolovém menu
time	hodnota datum a čas
array	řetězec hodnot uspořádaných do pole
nil	výchozí hodnota proměnné bez přiřazené hodnoty

2.3.1 Konstanty

Skriptovací jazyk umožňuje rovněž modifikaci textového řetězce *string*. Tyto úpravy zásadně mění chování výstupu při tisku.

Tab. 2.3: Speciální znaky

Znak	Význam
\”	vloží dvojité uvozovky - „“
\\	vloží zpětné lomítko - \
n	vloží nový řádek - *LF*
r	vloží posun na začátek řádku - *CR*
t	vloží horizontální tabulátor - *HT*
$	vloží znak dolaru - $, jinak je použit jako spojka proměnných
?	vloží znak otazníku - ?, slouží rovněž pro tisk konzolové nápovědy
_	vloží mezeru - „ “
a	vloží znak výstrahy - *BEL*
f	vloží znak posunu o stránku - *FF*
v	vloží vertikální tabulátor - *VT*
xx	vloží hexadecimální tisk znaku

2.4 Operátory

2.4.1 Aritmetické

Aritmetické operátory řídí vztahy mezi prvky ve výrazech. Skriptovací jazyk využívá běžné aritmetické operátory.

Tab. 2.4: Aritmetické operátory

Operátor	Operace	Vstup	Výstup
+	součet	(3 + 4)	7
-	rozdíl	(3 - 4)	-1
*	součin	(3 * 4)	12
/	podíl	(10 / 2)	5
-	negace	-(10)	-10

2.4.2 Relační

Relační operátory jsou použity k porovnání dvou prvků ve výrazu. Výsledkem operace je logická 1 nebo 0 (pravda nebo nepravda).

Tab. 2.5: Relační operátory

Operátor	Operace	Vstup	Výstup
<	menší než	(3 < 4)	pravda
>	větší než	(3 > 4)	nepravda
=	rovnost	(3 = 4)	nepravda
<=	menší nebo rovno	(10 <= 2)	nepravda
>=	větší nebo rovno	(10 => 2)	pravda
! =	nerovnost	(10 ! = 2)	pravda

2.4.3 Logické

Logické operátory jsou použity k porovnání dvou výrazů.

Tab. 2.6: Logické operátory

Operátor	Operace	Vstup	Výstup
! **- not**	jedničkový doplňek	! pravda	nepravda
&& **- and**	logický součin	pravda && pravda	pravda
|| **- or**	logický součet	pravda || nepravda	pravda
in		1.1.1.1/32 in 1.0.0.0/8	pravda

Logické operátory *not*, *and*, *or* a *in* patří rovněž do kategorie klíčových slov a nemohou být použity jako názvy proměnných.

2.4.4 Bitové

Bitové operátory jsou použity k porovnání dvou výrazů. Logické operace se provádějí bit po bitu a smí být použity pouze pro datové typy IP adresa a číslo.

Tab. 2.7: Bitové operátory

Operátor	Operace	Vstup	Výstup
~	inverze bitů	~0.0.0.0	255.255.255.255
|	bitový **OR**, výsledkem operace je „0“ pokud jsou oba bity rovny nule, jinak „1“	(10 | 10)	10
^	bitový **XOR**, výsledkem operace je „0“ pokud jsou si oba bity rovny, jinak „1“	(10 ^ 10)	0
&	bitový **AND**, výsledkem operace je „1“ pokud jsou oba bity rovny „1“, jinak „0“	(10 & 10)	10
<<	bitový posun vlevo o zadaný počet bitů	(10 << 2)	40
>>	bitový posun vpravo o zadaný počet bitů	(10 >> 2)	2

2.4.5 Slučovací

Slučovací operátory jsou použity pro sloučení dvou a více řetězců.

Tab. 2.8: Logické operátory

Operátor	Operace	Vstup	Výstup
.	sloučí dva řetězce	„abc“ . „def“	„abcdef“
,	sloučí dvě pole nebo vloží prvek	({1, 2, 3} , 5)	1;2;3;5

Do řetězců lze vkládat obsahy proměnných bez slučovacích operátorů. Použitím znaku *$* v řetězcích je možné dovnitř vložit výrazy.

2.5 Proměnné

Skriptovací jazyk používá dva typy proměnných.

První z nich jsou globální, přístupné uvnitř každého skriptu, definovány klíčovým slovem *global*.

Druhou variantou jsou lokální, přístupné pouze v aktuálním bloku skriptu, definovány klíčovým slovem *local*.

Každá proměnná, mimo již vestavěných konstant, musí být deklarována před jejím užitím klíčovými slovy *local* nebo *global*. Nedeklarované proměnné jsou označeny jako nedefinované a způsobí chybu při překladu. Validními znaky ve jménech proměnných jsou písmena anglické abecedy a čísla. Pro vytvoření proměnné se jménem obsahujícím jiný znak, musí být uzavřena do dvojitých závorek. Jestliže je proměnná deklarována bez inicializace, její datový typ je nastaven na *nil*, v jiném případě je automaticky detekován a nastaven překladačem. Někdy je ovšem nutností provést konverzi mezi datovými typy. Pro tyto účely jsou použity konverzí příkazy. Je nutné dodržet rozlišování velkých a malých písmen.

2.6 Příkazy

2.6.1 Globální

Veškeré globální příkazy musí začínat znakem „:“, v opačném případě budou zpracovány jako proměnné. Globální příkazy jsou dostupné v každém menu a skriptu. Přehled příkazů viz tabulka A.1.

2.6.2 Menu

Příkazy dostupné ve většině podmenu.

Tab. 2.9: Přehled příkazů užitých v menu

Příkaz	Syntaxe	Význam
add	add <parametr>=<hodnota>	vytvoří nový prvek
remove	remove <id>	zruší vybranou položku
enable	enable <id>	povolí vybranou položku
disable	disable <id>	zakáže vybranou položku
set	set <id> <parametr>=<hodnota>	změní parametr vybrané položky
get	get <id> <parametr>=<hodnota>	získá parametr vybrané položky
print	print <parametr><parametr>=[<hodnota>]	vytiskne vybrané položky menu
export	export [soubor=<hodnota>]	exportuje konfiguraci z aktuálního menu a jeho podmenu. Jestliže je zadán parametr soubor, výstup bude uložen do souboru s příponou „.rsc“, jinak je výstup tisknut do konzole
edit	edit <id> <parametr>	edituje vybranou položku vestavěným textovým editorem
find	find <výraz>	najde položky výrazu

Posledním specifickým příkazem je příkaz *import*. Lze jej volat pouze v kořenovém adresáři a slouží pro import konfigurace ze zvoleného souboru, který byl vytvořen příkazem *export*.

Parametry tisku

Příkaz *print* umožňuje zvolit formátování a obsáhlost výstupu.

Tab. 2.10: Přehled parametrů tisku

Parametr	Význam
append	
as-value	tiskne výstup jako pole parametrů a jejich hodnot
brief	tiskne výstup se stručným popisem
detail	tiskne výstup s detailním popisem
count-only	tiskne pouze počet položek v menu
file	tiskne výstup do souboru
follow	vytiskne aktuální výstup a tiskne průběžně jeho změny, dokud není stisknuta kombinace kláves *ctrl - c*
follow-only	vytiskne výstup a tiskne průběžně pouze nové vstupy, dokud není stisknuta kombinace kláves *ctrl - c*
from	tiskne pouze dané položky
interval	průběžně tiskne výstup v určeném časovém intervalu, nelze použít s parametrem **follow**
terse	tiskne detailní výstup v kompaktním, strojovém formátu
value-list	tiskne položky pod sebe
without-paging	pokud se výstup nevleze do konzolového okna, nezastaví ale vytiskne všechny informace najednou
where	příkaz následovaný parametrem *where* je použit pro filtrování výstupu

Pro tisk může být zadáno více parametrů najednou, které ovlivňují celkový výstup.

2.7 Cykly a podmínky

Tab. 2.11: Přehled cyklů a podmínek užitých v RouterOS

Příkaz	Syntaxe	Význam
do..while	:do {<příkazy>} while=(<podmínky>)	provádí příkazy dokud nejsou splněny podmínky, cyklus proběhně vždy minimálně jednou
while	:while (<podmínky>) do={<příkazy>}	provádí příkazy dokud jsou podmínky splněny, narozdíl od předchozího případu, cyklus nemusí vůbec proběhnout
for	:for <proměnná> from=<číslo> to=<číslo> step=<číslo> do={<příkazy>}	provádí příkazy do zadaného počtu iterací
foreach	:foreach <proměnná> in=<pole> do={<příkazy>}	provádí příkazy pro každý záznam v poli
if	:if (<podmínky>) do={<příkazy>} else={<příkazy>}	pokud jsou dané podmínky pravdivé provádí se příkazy v bloku *do*, jinak v bloku *else*, který však nemusí být specifikován

2.8 Repositář

Obsahuje veškeré skripty vytvořené uživateli. Skripty mohou být spouštěny automatizovaně některou z událostí, pomocí volání jiného skriptu nebo manuálně v konzoli příkazem *run*. Mezi události spouštějící skripty patří System Scheduler, který umí volat programy v určitých daných intervalech. Další možností je Traffic Monitor Tool, která spouští skripty v případě, kdy na zvoleném rozhraní dojde k překročení povoleného limitu přenesených nebo odeslaných dat za vteřinu. Posledním nástrojem je Netwatch Tool spouštějící skripty v případech, kdy dojde k připojení nebo odpojení sledovaného hostitele pomocí příkazu *ping*. Citováno dle [11].

3 ŘEŠENÍ

3.1 Analýza systému RouterOS

Operační systém RouterOS umožňuje využití vícero možností, jak jednotlivá data získat a jakým způsobem je interpretovat.

Jednou z možností je využití protokolu SNMP (*Simple Network Management Protocol*). Jedná se o protokol, sloužící k potřebám správy sítí, umožňující jednoduchou práci pro administrátory a uživatele. Systém RouterOS podporuje SNMP verze 2, která je navíc doplněna o autentizaci a zamezuje neoprávněným přístupům k zařízení. Pomocí tohoto protokolu lze snadno realizovat sběr vybraných dat a jejich další zpracování jako jsou například tabulky a grafy. Nevýhodou SNMP je nutnost použití dvou systémů a to strany monitorované a monitorovací. Monitorovaná strana, kterou je přístupový bod, shromažďuje informace o stavu systému a monitorovací, vznášející požadavky o zaslání konkrétních zpráv. Je tedy nezbytné použít další zařízení, například počítač.

Dalším způsobem je použití vestavěného nástroje pro monitorování jednotlivých rozhraní prostřednictvím protokolu HTTP (*Hypertext Transfer Protocol*), jednoduchou implementací ve webovém prohlížeči. Aktivací monitorování sítě pro dané rozhraní je spuštěn přístup ke grafům, znázorňujícím počet celkových přenesených dat za jednotku času. Jednoduchost používání tohoto nástroje přináší řadu nevýhod. Významnou nevýhodou je především obnovovací interval hodnot grafů, který nemůže být snížen pod hodnotu pěti minut a rovněž nelze sledovat tok dat pro jednotlivé připojené klienty.

Posledním a zvoleným postupem je vytvoření vlastního programu (skriptu) běžícího na přístupovém bodě. Skripty mohou obsluhovat prakticky veškeré interní funkce systému a dovolují uživatelům automatizovaně spravovat nastavení zařízení, bez nutnosti vnějšího zásahu.

3.2 Problematika vytváření skriptu

Tato sekce se zabývá vzniklými problémy při vytváření skriptu. Přehlednější řešením problému je vytvoření vývojového diagramu. S jeho pomocí lze snadněji rozpoznat případné chyby, které mohou nastat při implementaci.

Základním problémem vývoje skriptu je počet připojených klientů. Jelikož je počet připojení předem neznámý, je bráno v potaz, že program musí být více obecnější. Navíc je nevýhodou realizace programu samotný sběr dat, poněvadž jsou vybraná data ukládána do souborů, což má za následek prodloužení času, potřebného pro

jeho vykonání. Soubory totiž nelze vytvářet přímo. Musí být použito funkcí pro export nastavení nebo tisk do souboru, lišících se pouze v použité příponě. Všechny vytvořené soubory jsou uloženy v adresáři file. V programu je použito funkce tisku, jejíž hlavní výhodou je vytvoření souboru textového.

Zásadní problematikou je rychlost zápisu dat do flash paměti. Doba trvání zápisu je přibližně půl vteřiny. Toto je dáno i samotnou funkcí pro tisk, která do souboru následně zbytečně vypisuje aktuální data pro dané konzolové podmenu. Možným způsobem, jak zkrátit dobu zápisu, je provést tisk v podmenu, které obsahuje nejméně informací nebo s dalším parametrem tisku, ačkoli tento způsob není zásadní. Jiným, přijatelnějším způsobem je dopředné vytvoření souborů, avšak lze se omezit pouze na povolené připojené klienty, jejich fyzickou adresou. V tomto případě nemusí program soubory vytvářet a operace zápisu je značně urychlena. Tento jev je zejména důležitý pro skripty spouštěné časovačem, který se naneštěstí nestará o skončení skriptu a v daném intervalu jej opět spouští. Proto může dojít k přepsání stávájících uložených hodnot, což je nepřípustné. Lze však časovač přizpůsobit pravděpodobnosti doby trvání běhu programu a výsledek průměrovat nastaveným intervalem, což je v práci uvažováno.

Dalším omezením programu je rovněž nemožnost zápisu a čtení dat v souboru, jehož velikost je větší než 4096 kB. Jelikož důvod tohoto omezení je neznámý, musí být po dosažení této hodnoty vytvořen nový soubor, což vede k problému popsanému výše. Možným opatřením je odeslání plného souboru prostřednictvím FTP (*File Transfer Protocol*) protokolu nebo pomocí elektronické pošty. Tato implementace však není provedena.

Dalším problémem je přesun vytvořených souborů. Možným postupem je použití výše uvedeného protokolu FTP nebo elektronické pošty. Pro dané účely je ovšem vybrán program WinBox, vyvinutý vývojáři RouterOS, umožňující řízení interních funkcí přístupového bodu pro operační systémy, založené na platformě Windows. Zde nastává problém při přesunu souborů. Operační systémy Windows neumožňují pojmenování souborů, obsahující speciální řídící znaky, mezi které patří i dvojtečka, obsažena ve fyzické adrese klienta. Název vytvořeného souboru obsahuje fyzickou adresu sledovaného klienta. Z tohoto důvodu musí být změněn název souboru, kde jsou data uložena nebo obsah souboru před exportem kopírovat do souboru s jiným názvem.

Poslední důležitou části je ladění programu. Skriptovací jazyk nezastává možnosti, jakými hledat chyby a nedostatky ve skriptech. Komentáře, které lze vpisovat do kódu, mají význam pouze informativní a slouží pro přehled jednotlivých funkcí programu a tudíž nenabývají velkého významu. Další možností je použití vestavěného reproduktoru v zařízení. V určitém úseku kódu lze vložit příkaz pípnutí. Při spuštění programu tento zvuk napoví, zda-li je dosaženo daného bodu programu.

Rovněž může být použito editace skriptu v konzoli, kdy je při jeho kompilaci oznámeno, na kterém řádku a sloupci se daná chyba vyskytuje. Při spuštění je tato chyba vypsána do konzole.

3.3 Implementace zdrojového kódu

Jak již bylo popsáno, sběr dat je na přístupovém bodě prováděn pomocí samotného skriptu. Skript musí být koncipován tak, aby jeho časová složitost a paměťová náročnost příliš nevytěžovala přístroj a neomezovala jeho funkčnost.

V prvém kroku je pro zjednodušení vyvoláno menu registrační tabulky bezdrátového rozhraní. Využití této možnosti přináší značné usnadnění. Získání údajů připojených klientů nevyžaduje u příkazů relativní cestu k podmenu.

```
/interface wireless registration-table {
```

Je nutností ověřit, zda je k přístupovému bodu připojen alespoň jeden klient, protože v jiném případě nemohou být data získána. Toto lze provést pomocí příkazové podmínky. Je ovšem nutné, v prvém kroku, deklarovat příslušné proměnné a to pomocí klíčového slova global nebo local. Je použita globální proměnná a to z důvodu její existence po skončení skriptu. Pokud by nastala běhová chyba, lze snáze rozpoznat, zda-li jsou vybrány správné hodnoty, které zůstanou ve vytvořených proměnných uloženy.

```
:global pocet_klientu [:len [find]]
:if ($pocet_klientu > 0) do={
```

Nyní je provedena okamžitá inicializace proměnné v níž je uložena hodnota počtu klientů.

Dalším krokem je cyklus, který proběhne pro každého klienta v registrační tabulce. Použity mohou být cykly dva a to cyklus *while* nebo *for*.

```
:while ($krok_cyklu < $pocet_klientu) do={
:for $krok_cyklu from=0 to=($pocet_klientu - 1) do={
```

Lepším řešením je použití cyklu *for*. Důvod tohoto počínání tkví v tom, že cyklus *while* vyžaduje pro své provedení předem deklarovanou proměnnou *krok_cyklu*, avšak cyklus *for* nikoliv.

Nyní je ze seznamu parametrů klienta vybrána jeho fyzická adresa a objem celkových přenesených dat. Data jsou uložena do proměnných *MAC* a *hodnota*. Toto je provedeno pomocí příkazu *get*.

```
:set MAC [:tostr [get number=$krok_cyklu mac-address]
:set hodnota [:toarray [get number=$krok_cyklu bytes]
```

Datové typy proměnných jsou rozdílné. Proměnná *MAC* obsahuje hodnotu typu řetězec a proměnná *hodnota* je typu pole. Důvod tohoto počínání je v praktičnosti. Vzhledem k tomu, že hodnoty uchovávané pro každou vlastnost klienta jsou interpretovány příkazem *print* jako řetězec znaků (konkrétně vlastnost *bytes*, obsahující objem přenesených dat, je ve formátu „přijatá,odeslaná“ data), je použit příkaz *toarray* k jejich rozdělení do pole. Hodnota na indexu 0 představuje data přijatá, na indexu 1 pak data odeslaná.

Poněvadž jsou data ukládána do souborů, je nutné ověřit, zda-li je pro vytvoření nebo zápis dat do souboru, volné místo ve flash paměti.

```
:if ([:tonum [/system resource get free-hdd-space]] > 2048) do={
```

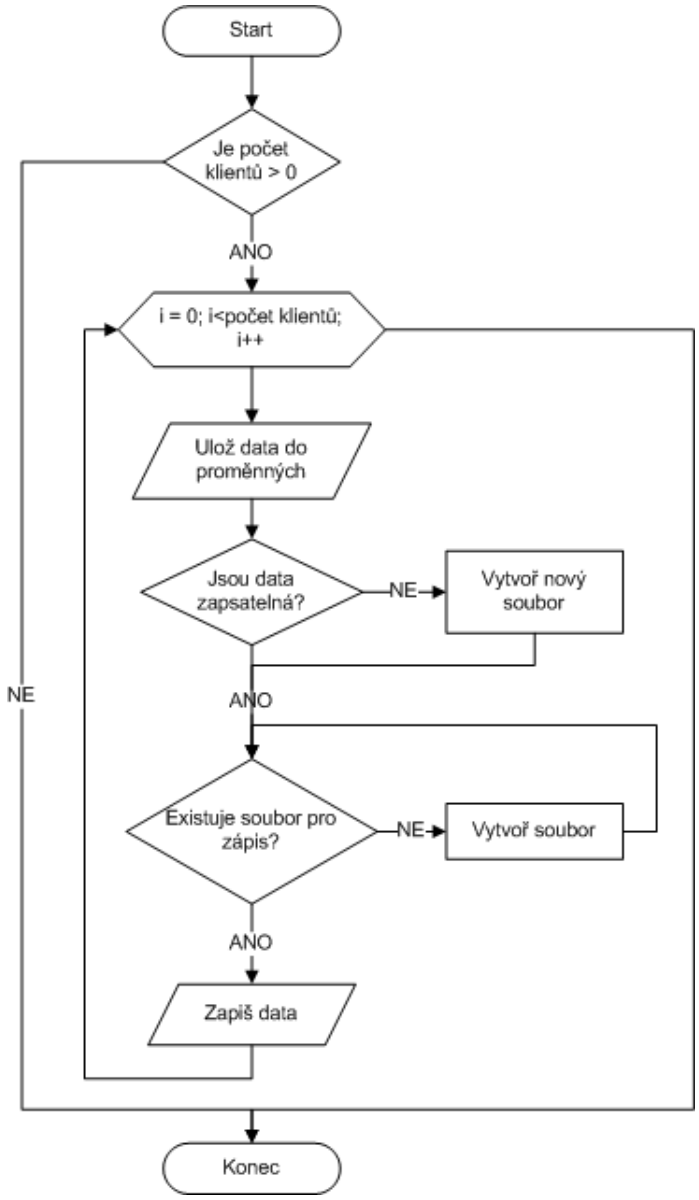
Z hlediska volného místa v paměti je vždy ponechána rezerva ve velikost 2 MB pro případné další možné operace.

Jako názvy jednotlivých souborů je použita fyzická adresa klienta ve tvaru „Bytes-MAC“ a „RxTxX-MAC“, kde X je číslo souboru a *MAC* fyzická adresa klienta. Soubor „Bytes-MAC“ obsahuje počet celkových přenesených dat a soubor „RxTxX-MAC“ rozdíl mezi nově získanými přenesenými daty a daty předchozími za daný časový úsek.

Ve své podstatě není nutné vytvářet soubor pro ukládání objemu celkových přenesených dat, jelikož lze data uložit do globální proměnné. Nicméně toto platí pouze v případě jediného klienta. Počet klientů není dopředu znám a je očekáváno vícero připojení, proto musí být celková přenesená data všech připojených klientů uchována v souborech.

Další krok představuje zjištění, jestli již existuje soubor pro uložení dat daného klienta. Pokud ne, je nutno soubor vytvořit a data do něj zapsat. V jiném případě je nalezen poslední vytvořený soubor a ověřeno, zda-li lze data zapsat. Tento krok je velmi důležitý, jelikož jsou skripty omezeny limitem velikosti souboru pro čtení a zápis dat a jsou schopny ukládat a číst data o maximální velikosti 4096 bajtů, což je rovno stejnému počtu znaků. Pokud je součet velikostí výsledných dat a obsahu souboru větší než povolený limit, musí být vytvořen nový soubor. Pokud jsou však data uložitelná, je proveden zápis.

Poslední důležitou částí programu je jeho spouštění. Pro tento úkon je zaveden vestavěný plánovač úloh, který umožňuje spouštění skriptu v daných časových intervalech zadaných v sekundách. Dle nastaveného intervalu lze tedy průměrovat počet přenesených dat za jednotku času a v případě dalšího zpracování vynést tyto údaje do grafů nebo použít pro další zpracování. Nelze však zapomenout na dostatek volného místa v paměti pro uložení nových dat. V případě jeho nedostatku je nutné systémový plánovač zastavit, což uvolní systémové prostředky. Pro názornou ukázku je přiložen vývojový diagram viz níže.



Obr. 3.1: Vývojový diagram

Pro interpretaci souborů přístupového bodu lze použít program WinBox, který umožňuje jejich stažení pro operační systémy, založené na platformě Windows. Jiný způsob umožňuje odeslání souborů pomocí elektronické pošty nebo FTP protokolu.

4 ICMP PŘENOS

4.1 Metarouter a OpenWRT

Jak již bylo popsáno v předchozí části práce, metarouter je možnost operačního systému RouterOS vytvořit virtuální směrovač, ke kterému lze připojit mimo výchozího systému i jiný operační systém, kterým je v tomto případě OpenWRT. RouterOS rovněž umožňuje připojit k virtuálnímu zařízení i fyzické rozhraní, proto může virtuální systém doplnit nebo případně plně nahradit hostující RouterOS. Co se týče ztráty výkonu, je tato hodnota spekulativní, jelikož pro každý metarouter, kterých může být až osm, je přiřazeno maximální využití paměti a místa na disku a tyto hodnoty mohou být i zpětně změněny. Jelikož je RouterOS velmi schopným systémem založený na Unixovém jádře, je virtualizované OpenWRT ideální volbou pro jeho doplnění ve formě tvoření vlastních programů pro práci v síti. Základním faktorem pro uskutečnění samotné virtualizace je RouterBoard RB400, RB700, RB800 nebo RB1000 s dostatečně novou verzí RouterOS.

Prvním důležitým krokem pro vytvoření virtualizovaného OpenWRT, je získání jeho obrazu. Jednou z možností je stáhnutí obrazu z fóra Mikrotiku nebo získání systému přímo ze stránek OpenWRT a aplikovat na něj doplňkový balíček, který je rovněž ke stažení na [4]. Použití balíčku je nutností, jelikož systém, distribuovaný samotnými vývojáři, nepočítá s jeho použitím ve virtualizované formě. Podmínkou jeho aplikace je provedení kompilace jádra pro MikroTik metarouter.

4.2 Systém a kompilace

Kompilace jádra je prováděna na operačním systému linuxové distribuce Debian verze 6. Důvodem je množství balíčků uvedených v závislostech na OpenWRT, které jsou pro kompilaci použity. Pro plně funkční operační systém postačí stáhnutí jeho prvního obrazu ve formátu DVD. Ostatní obrazy obsahují jen další balíčky pro danou distribuci.

Po instalaci systému jsou veškeré operace prováděny v terminálu, ovšem bez práv správce. Ten lze najít v systémovém menu Aplikace → Příslušenství → Terminál. Pokud jsou ovšem pro vykonání příkazu práva nezbytné, musí být příkazy spuštěny s předponou *sudo*.

Nejprve je nutné nainstalovat na hostitelském PC obecné balíčky pro vývoj aplikací (C/C++) build-essential a subversion. Instalace je provedena pomocí následujících příkazů:

sudo apt-get update	obnovení seznamu dostupných balíčků pro danou distribuci
sudo apt-get install build-essential subversion	instalace balíčků

Dalším krokem je instalace nezbytných balíčků pro chod OpenWRT pomocí příkazu:

sudo apt-get install asciidoc binutils bzip2 fastjar flex g++ gcc autoconf gawk bison libgtk2.0-dev intltool jikes zlib1g-dev make libncurses5-dev libssl-dev patch perl-modules rsync ruby sdcc unzip wget sdcc-nf gettext xsltproc zlib1g-dev

Jestliže by některý z balíčků pro daný systém nebyl k dispozici, lze jej přeskočit, protože při výsledné kompilaci systém kontroluje, zdali jsou všechny potřebné balíčky k dispozici. Pokud tomu tak není, sám se přihlásí o balíčky, které chybí a ty lze vyhledat a dodatečně nainstalovat. Pomocí příkazu subversion je provedeno stažení vývojové verze OpenWRT:

svn co svn://svn.openwrt.org/openwrt/trunk
cd trunk

Následující krok je aplikace balíčku staženého ze stránek MikroTiku:

wget http://www.mikrotik.com/download/openwrt-metarouter-1.2.patch
patch -p0 < openwrt-metarouter-1.2.patch

Po úspěšné aplikaci, je třeba spustit konfigurační menu příkazem:

make menuconfig

V konfiguračním menu nastavit *Target System* na „Mikrotik MetaROUTER“. Zbylé nastavení záleží jen na způsobu využití systému. V menu *toolchain-options* je však nutností přidat i kompilátor *gcc*, bez něhož nelze tvořit programy. Po zkompilování je systém připravený v souboru *openwrt-metarouter-rootfs.tgz*.

4.3 Instalace do RouterOS

Pro instalaci OpenWRT na RouterOS je potřeba zkompilovaný systém přenést do přístupového bodu. To lze provést pomocí FTP protokolu (nutný je však přenos v binárním režimu) nebo programu WinBox. Pro přehlednost je doporučen program WinBox, kde po přihlášení do směrovače lze soubor zkopírovat přetažením do složky *files*. S pomocí konzole na RouterOS se použitím příkazu *import-image* nainstaluje připravený systém. Povinným argumentem příkazu *import-image* je název souboru *file-name*. Zpracovávaný soubor musí obsahovat příponu *.tgz*. Jelikož jde o komprimovaný soubor, nesmí jeho velikost před dekomprimací přesáhnout polovinu volného místa na disku.

Dalšími nepovinnými argumenty mohou být *disk-size* – přidělené místo na disku udáno v kB (implicitně bez limitu), *memory-size* – přidělená velikost paměti udána v MB (implicitně 16 MB) a *name* – jméno virtuálního zařízení (implicitně mr$X$, kde X je číslo v pořadí virtuálního zařízení).

Pokud je použito základní nastavení pro využití disku, čili neomezené, hrozí neočekávané nebezpečí jeho zaplnění ze strany virtuálního systému.

```
[admin@MikroTik] /metarouter> import-image file-name=openwrt-mr-mips-rootfs.tgz
 imported: 100%
```

```
[admin@MikroTik] /metarouter> print
Flags: X - disabled
#   NAME      MEMORY-SIZE DISK-SIZE     USED-DISK     STATE
0   mr1       16MiB       unlimited     7383kiB       running
```

Obr. 4.1: Použití příkazu import-image

Příkazem *print* se zobrazí nainstalovaný systém, velikost přidělené paměti a využití místa na disku. Nyní je potřeba, pomocí příkazu *add*, přidat nové rozhraní k virtuálnímu směrovači v menu *interface*. Na fyzickém směrovači se následně objeví rozhraní jako dynamické.

```
[admin@MikroTik] /metarouter> interface add
comment    disabled        dynamic-mac-address  type            virtual-machine
copy-from  dynamic-bridge  static-interface     vm-mac-address

[admin@MikroTik] /metarouter> interface add virtual-machine=mrl type=dynamic

[admin@MikroTik] > /interface print
Flags: D - dynamic, X - disabled, R - running, S - slave
#     NAME                                                TYPE            MTU
8  R  ether9                                              ether           1500
9  R  test                                                bridge          1500
10 DR vif1                                                vif             1500
```

Obr. 4.2: Přidání virtuálního rozhraní

Po vytvoření virtuálního rozhraní je nutné jej spojit s rozhraním fyzickým. Připojení virtuálního rozhraní k fyzickému rozhraní je provedeno pomocí přemostění. K vytvořenému mostu, na hostitelském systému, jsou navázána fyzická a virtuální rozhraní (v tomto případě OpenWRT), což umožní průchod síťovému provozu a vzájemné komunikaci.

Jelikož je OpenWRT linuxová distribuce operačního systému, je příkazově prakticky totožná s běžnými linuxovými systémy s rozdílem zmenšeného jádra. Znamená to, že disponuje pouze vybranými bash příkazy, jejíž příčinou je omezená velikost systému v rámci náročnosti na daná zařízení. Citace [4].

4.4 Nastavení OpenWRT

Konfigurace OpenWRT probíhá pomocí konzole. V prvním kroku je nutné spustit systémovou konzoli metarouteru z konzole MikroTiku. To lze provést v menu *metarouter* pomocí příkazu „**console** *name*“, kde *name* je název vytvořeného metarouteru. Po použití příkazu se načte konzole OpenWRT a po stisknutí klávesy *ENTER* je dosáhnuto kořenového adresáře. V adresáři *etc/config*, pomocí editoru *vi*, je možné měnit konfigurační soubor *network*, kde jsou obsažena veškerá síťová nastavení. Lze zde použít dynamické získávání nastavení nebo provést statickou konfiguraci. Pro aktivaci změn je nutné v adresáři *etc/init.d* spustit soubor **network** s parametrem *restart* nebo *reload*.

5 VÝVOJ PROGRAMU

Vytvoření programu je možné jen na operačním systému s linuxovým jádrem. Pro tvorbu programu je použit operační systém Debian verze 6. Výhodou je vestavěný textový editor *gedit*. Umožňuje zejména zvýrazňování syntaxe pro jazyk C a tím zjednodušuje a zpřehledňuje psaní zdrojového kódu.

5.1 Struktury

Bakalářská práce se zabývá možností přeposílání uživatelských zpráv pomocí protokolu ICMP. ICMP paket sám o sobě obsahuje hlavičku, která je složena z typu, kódu a kontrolního součtu. Dále však může rovněž nabývat v případě echo žádostí i identifikační číslo, sekvenci a další větve unie (popsané níže) v závislosti na zvoleném typu a kódu ICMP paketu. Za touto hlavičkou následují data, která zde však být nemusí. Tyto data mohou obsahovat uživatelem definovaný obsah. To je pro tuto práci velmi významné. Zvolená data v tomto případě potřebují pouze hlavičku ICMP paketu, jehož struktura je obsažena v knihovně *ip/ip_icmp.h* a v jazyce C je následující:

```
struct icmphdr
{
  u_int8_t type;                /* typ zprávy */
  u_int8_t code;                /* kód typu */
  u_int16_t checksum;           /* kontrolní součet */
  union
  {
    struct
    {
      u_int16_t id;             /* id procesu */
      u_int16_t sequence;       /* sekvenční číslo */
    } echo;                     /* echo datagram */
    u_int32_t gateway;          /* adresa brány */
    struct
    {
      u_int16_t __unused;       /* rezervováno */
      u_int16_t mtu;            /* maximální velikost paketu */
    } frag;                     /* fragmentace */
  } un;
};
```

Typ zprávy určuje druh ICMP zprávy. Jedná se zejména o informační nebo chybová hlášení, která mohou být pro některé typy dále upřesněna pomocí typového kódu.

Kontrolní součet snižuje pravděpodobnost výskytu chyby při přenosu zprávy. Obsahuje součet velikostí všech datových typů v hlavičce ICMP zprávy a případných dat, která jsou za hlavičkou obsažena.

ID procesu identifikuje program, který danou zprávu odeslal. V jakémkoliv operačním systému je každý proces identifikován svým jednoznačným číslem. Není nutné jej používat, avšak v případě žádosti je pro zajištění totožnosti mezi odeslanou a přijatou zprávou nezbytné jej použít.

Sekvenční číslo je použito pro zprávy typu žádost (*echo*). Jde například o program ping, který za sebou posílá jednotlivé pakety počínaje číslem 0 a do každé následující zprávy je sekvenční číslo inkrementováno o 1. Příjemce pak přepíše typ zprávy na odpověď (*reply*) a pošle ji zpět jinak nezměněnou odesílateli. Ten pak díky sekvenčnímu číslu vyhodnotí, zda-li nebyl v síti některý z paketů ztracen.

Adresa brány (gateway) je použita u zprávy typu 5 (*redirect*) pro přesměrování. Měla by být generována přímo bránou, na které je zjištěn požadavek na přesměrování spojení.

MTU (*Maximum Transmission Unit*) určuje maximální velikost paketu, který je přenesen mezi dvěma body, bez použití fragmentace. Obvyklá hodnota MTU u technologie Ethernet je 1500 bajtů, může se však v určitých místech počítačové sítě měnit, například ve spojení se sériovou linkou, kde může být hodnota MTU nižší. Pokud některý ze směrovačů na cestě v síti přijme datagram s nastaveným příznakem nefragmentovat, avšak pro další pokračování je nutné fragmentaci provést, odešle ICMP oznámení o chybě. To platí jen v IPv4, jelikož při ztrátě jediného fragmentu je nutné přenášet celý chybějící celek, což působí potíže zejména u vyšších protokolů. Proto IPv6 místo fragmentace příliš velké pakety zahazuje. Citace dle [9].

Výsledná struktura ICMP paketu i s daty, která se budou přenášet, mají tuto konstrukci:

```
struct icmpdata
{
  struct icmphdr icmp;
  u_int16_t used;          /* data */
  u_int32_t count;
};
```

Struktura může nabývat i jiné nebo další hodnoty samozřejmě s ohledem na velikost výsledného paketu. Důležitý je však výpočet kontrolního součtu z celé struktury,

protože některá zařízení v síti mohou při jeho nesprávném výpočtu paket zahodit nebo odeslat chybové hlášení.

Síťová adresa je definována strukturou:

```
struct in_addr {
  unsigned long s_addr;   /* Síťová adresa */
};
```

Posílání datagramů v síti je provázeno síťovými adresami. Pro převod IP adresy v tečkovém formátu na síťovou adresu je použita funkce:

```
int inet_aton(const char *cp, struct in_addr *inp);
```

kde parametr *cp* vyjadřuje IP adresu a parametr *inp* cílovou strukturu pro uložení převedené adresy. Funkce vrací v případě úspěchu jakékoliv nenulové číslo, v opačném případě 0.

Jelikož je v programu použito skupinové vysílání, vyžaduje následující strukturu:

```
struct ip_mreq
{
  struct in_addr imr_multiaddr;   /* Multicastová adresa skupiny */
  struct in_addr imr_interface;   /* Adresa rozhraní */
};
```

Struktura je obsažena v knihovně *netinet/in.h*.

Je použita v kombinaci s přihlašováním do multicastové skupiny viz kap. 5.2.1. Proměnná *imr_multiaddr* obsahuje síťovou adresu skupiny, kam má být rozhraní, dané proměnnou *imr_interface*, přihlášeno.

Odesílání dat vyžaduje jednoznačné určení příjemce zprávy. Problematiku řeší struktura:

```
struct sockaddr_in {
    short            sin_family;   /* Doména */
    unsigned short   sin_port;     /* Port */
    struct in_addr   sin_addr;     /* Síťová adresa */
    char             sin_zero[8];  /* */
};
```

Proměnná *sin_family* identifikuje komunikační doménu. Z hlediska programu se jedná o makro AF_INET. Proměnná *sin_port* pak port, na kterém přenos probíhá. Není použit a je nastaven na hodnotu 0. Struktura *sin_addr* viz kapitola 5.1.

5.2 Sockety

Pro odesílání a přijímání dat je v programu použito socketů. Sockety jsou univerzálním komunikačním prostředkem. Jedná se vlastně o jeden z prostředků meziprocesní komunikace. Od jiných komunikačních prostředků se liší především tím, že komunikující procesy nemusí být na stejném počítači a lze je tudíž použít v síti. Konstrukce socketu je následující:

```
int socket (int domain, int type, int protocol);
```

kde *domain* značí komunikační doménu, na které je socket vytvořen. Domény jsou rozděleny na 2 části a to protokolovou PF (*Protocol Family*) a adresní AF (*Address Family*). Rozdíl mezi těmito rodinami tkví v tom, že rodina PF identifikuje soubor protokolů se stejnou architekturou a AF se stejným formátem adres. Podle [10] pozdější standardy určují obě rodiny za obecně stejné a používají pouze rodinu AF. Nicméně pro zvolený program je z důvodu aplikace protokolu ICMP použito rodiny PF, pro niž platí makro PF_INET.

Argument *type*, určuje komunikační styl socketu. Mezi základní patří typ spojovací SOCK_STREAM (v prostředí sítí odpovídá protokolu TCP), datagramový SOCK_DGRAM (v prostředí sítí odpovídá protokolu UDP) a čistý SOCK_RAW. RAW socket, narozdíl od ostatních typů, umožňuje přímý přístup k protokolu IP a zároveň při přijímání dat bere v potaz i hlavičky paketů, nikoliv jen užitná data. V programu je tedy použito makro SOCK_RAW.

Argument *protocol* určuje specifický protokol, který bude použit pro odesílání či přijímání dat. V tomto případě makro IPPROTO_ICMP.

Funkce vrací file-descriptor, který identifikuje vytvořený socket a je použit u funkcí pracujícími se socketem. Viz [5].

V knihovně *sys/socket.h* jsou vytvořeny globální konstanty usnadňující nastavení rodiny, typu i protokolu daného socketu.

5.2.1 Nastavení socketu

Nedílnou součástí komunikace mezi přístupovými body je nastavení voleb socketu. Pro tyto účely je zavedana funkce:

```
int setsockopt(int s, int level, int optname, const void *optval,
               socklen_t optlen);
```

Prvním parametrem je identifikátor socketu, vytvořený pomocí funkce **socket**. Druhým parametrem je úroveň volby. Třetím parametrem je název volby. Tyto dva parametry přesně určují atribut, jehož hodnota má být nastavena. Čtvrtým parametrem je ukazatel na blok paměti, ve které se nachází zadávaná hodnota. Posledním

parametrem je velikost alokované paměti, na kterou se odkazuje parametr *optval*. Funkce vrací 0 v případě úspěchu. V opačném případě –1.

Parametry socketu lze měnit na různých vrstvách síťového modelu. Parametr *level* funkce **setsockopt** určuje, na které vrstvě se daná vlastnost nachází. Řešení programu však používá pouze volby protokolu IP, pro který je definováno makro IPPROTO_IP.

Program využívá pro odesílání zpráv skupinové vysílání. Parametr *optname* tudíž vyžaduje nastavení jeho maker. Dle [13] jsou následující:

- **IP_MULTICAST_LOOP** definuje, zda-li má být odeslaná zpráva opět přijata zdrojem vysílání nebo nikoliv. Jelikož je program navržen pro přijímání i odesílání dat, je tento jev nežádoucí. Vypnutí smyčky je charakterizováno nastavením parametru *optval* funkce **setsockopt** na 0.
- **IP_MULTICAST_TTL** definuje TTL (*Time To Live*) zprávy. Pokud není zadáno jinak, je implicitně hodnota nastavena na 1. Tím je zabráněno směrování datagramů mimo lokální síť.
- **IP_MULTICAST_IF** definuje rozhraní, na němž bude komunikace probíhat. Parametr *optval* přebírá adresu IP, která je rozhraní přiřazena. Není-li uvedeno jinak, je použito výchozí rozhraní.
- **IP_ADD_MEMBERSHIP** definuje připojení rozhraní k multicastové skupině. Parametr *optval* přebírá strukturu **ip_mreq**.

5.3 Práce s daty

5.3.1 Odesílání

Odesílání dat probíhá pomocí funkce:

```
int sendto(int sockfd, const void *msg, int len, unsigned int flags,
           const struct sockaddr *to, socklen_t tolen);
```

Prvním parametrem je identifikátor socketu, vytvořený pomocí funkce **socket**. Druhým parametrem je požadovaná zpráva, tedy naplněná struktura **icmpdata**. Parametr *len* označuje velikost vstupních dat. Parametr *flags* specifikuje typ přenosu zprávy. Není použit a je nastaven na hodnotu 0. Parametr *to* označuje cílovou adresu, uloženou ve struktuře **sockaddr**. Vzhledem k přehlednosti je použita struktura **sockaddr_in**, která je uživatelsky jednodušší. Posledním parametrem je velikost parametru *to*. V jazyce C lze použít pro výpočet velikostí datových typů vnořenou funkci **sizeof**().

Funkce vrací v případě úspěchu počet odeslaných bajtů. Při výskytu chyby -1.

ICMP paket nikdy nemůže stát samostatně. Vždy je odeslán spolu s hlavičkou IP, která identifikuje zdrojovou stanici. Jelikož je socket vytvořen na úrovni SOCK_RAW, je před ICMP zprávu automaticky vložena hlavička IP operačním systémem.

5.3.2 Příjem

Příjem dat probíhá pomocí funkce:

```
int recvfrom(int sockfd, void *buf, int len, unsigned int flags,
             struct sockaddr *from, int *fromlen);
```

Prvním parametrem je identifikátor socketu, vytvořený pomocí funkce **socket**. Druhým parametrem je ukazatel na blok paměti, kde mají být přijatá data uložena. Parametr *len* označuje velikost alokované paměti parametru *buf*. Parametr *flags* specifikuje typ přenosu zprávy. Není použit a je nastaven na hodnotu 0. Parametr *from* označuje ukazatel na strukturu **sockaddr**, kde mají být zaznamenány identifikátory vzdálené stanice. Posledním parametrem je velikost parametru *from*.

Funkce vrací v případě úspěchu počet přijatých bajtů. Při výskytu chyby -1.

Funkce **recvfrom** ovšem nekontroluje, zda-li jsou data na socketu k dispozici. Vzhledem k tomuto faktu je zavedena funkce **select** a makra s ní pracující:

```
int select(int n, fd_set *readfds, fd_set *writefds,
           fd_set *exceptfds, struct timeval *timeout);

struct timeval {
  long tv_sec;      /* Sekundy */
  long tv_usec;     /* Milisekundy */
};

FD_SET(int fd, fd_set *fdset);
FD_CLR(int fd, fd_set *fdset);
FD_ISSET(int fd, fd_set *fdset);
FD_ZERO(fd_set *fdset);
```

Makro **fd_set** je použito jako název datového typu. Jedná se o typ množina.

- **FD_ZERO** makro sloužící k vyprázdnění množiny.
- **FD_CLR** makro sloužící k odebrání prvku z množiny. Prvním parametrem je identifikátor socketu. Druhým parametrem pak ukazatel na proměnnou typu **fd_set**.
- **FD_SET** makro sloužící k vložení prvku do množiny.
- **FD_ISSET** makro sloužící k otestování množiny na přítomnost prvku. Rozbalí se na výraz, který vrací 0 v případě, že prvek v množině není. V opačném případě vrátí jakoukoliv nenulovou hodnotu.

První parametr funkce **select** je socket s nejvyšším identifikátorem inkrementovaný o 1. Druhý parametr je ukazatel na množinu obsahující sockety, z nichž je čteno. Parametry *writefds* a *exceptfds* nejsou programově významné a jejich hodnota je NULL. Posledním parametrem je ukazatel na strukturu **timeval**, která obsahuje interval, který funkce **select** čeká na přítomnost změny v množině **fd_set**.

Před vlastním voláním funkce **select** je použito maker **FD_ZERO** a **FD_SET**. Jelikož program využívá pouze jediný socket, bude množina obsahovat pouze jeden prvek. Pokud funkce během časového limitu zjistí změnu na socketu, změní hodnotu posledního parametru *timeout* na čas, zmenšený o hodnotu, kterou funkce **select** původně čekala. Zda-li se událost odehrála na použitém socketu je zjištěno pomocí makra **FD_ISSET**. Následně jsou pak data přijata pomocí funkce **recvfrom**. Příklady použití funkcí viz obrázky B.7 a B.8. Citováno dle [12].

5.4 Aplikace programu

Zdrojový kód programu je vytvořen v jazyce C/C++. K jeho kompilaci je použit kompilátor *mips-openwrt-linux-uclibc-g++*, který lze na Debianu nalézt, po výsledné kompilaci OpenWRT, v adresáři *trunk/staging_dir/toolchain-mips_gcc-linaro_uClibc-0.9.32/bin*. Veškeré použité knihovny pak v adresáři */include*. Kompilátor je spuštěn s parametry:

-Wall -pedantic -g *soubor* **-o** *výstup*

kde *soubor* obsahuje zdrojový kód programu pro přeložení a *výstup* název cílového přeloženého programu. Parametry *Wall* a *pedantic* zajišťují zobrazení veškerých varovných hlášení.

Dalším krokem je přesun programu do OpenWRT. To lze provést v terminálu, pomocí protokolu SFTP (*Secure File Transfer Protocol*) příkazem *sftp*. Nápovědu k příkazu lze získat jeho spuštěním bez parametrů.

Přesun probíhá ve dvou fázích. Prvním krokem je přesun souboru do RouterOS, jelikož implicitně pracuje jako FTP server. Následně pak použitím příkazu *sftp* v konzoli OpenWRT přesunout soubor z RouterOS. Tomuto lze předejít stáhnutím a spuštěním FTP serveru na straně Debianu a přímím přesunem souboru.

Po dokončení všech operací je nutné programu přiřadit práva správce. V jiném případě je program nespustitelný. To je provedeno pomocí příkazu:

chmod 7777 *soubor*

kde *soubor* označuje vytvořený program.

Před spuštěním programu jsou nutná nastavení fyzického zařízení, popsány v kapitole 6. Ovládání programu je popsáno v příloze C.

6 WDS

(*Wireless Distribution System*) je dle [6] systém umožňující bezdrátové propojení přístupových bodů v sítích standardu IEEE 802.11. Umožňuje rozšíření bezdrátové sítě za použití vícero přístupových bodů bez nutnosti tradičního požadavku na páteřní kabelové spojení. Výhodou WDS oproti jiným řešením je zachování MAC adresy klientských rámců přes spojení mezi přístupovými body.

WDS není zatím oficiálním standardem IEEE a proto je jeho použití na přístupových bodech různých výrobců odlišné. Toto může mít za následek jeho nefunkčnost, a proto je doporučeno používat zařízení stejného výrobce. WDS je možno použít ve dvou základních režimech a to přemostění nebo opakovač. Při přemostění je předáváno pouze spojení z jednoho přístupového bodu na druhý, není ale povoleno připojení klientů. Naopak při funkci opakovače je možné, aby distribuce probíhala nejenom na spojení mezi přístupovými body, ale rovněž poskytovala své služby klientům.

Nevýhodou WDS je připojení přístupových bodů a klientů ve stejném frekvenčním pásmu. To vede ke snížení propustnosti a zvýšení zarušení signálu. Rychlost na každém skoku bezdrátového opakovače je zhruba polovinou rychlosti předchozího skoku. Viz [7].

Na MikroTiku je však situace mírně odlišná. WDS pracuje na nejnižší základní rychlosti. To znamená, že neovlivňuje příliš propustnost připojených klientů a rovněž snižuje možnost výskytu chyb. Dále podporuje zabezpečení spojení mezi jednotlivými stanicemi pomocí dynamických klíčů. Navíc mohou jednotlivé přístupové body používat pro komunikaci jiný šifrovací algoritmus a klíč než spojení klient – přístupový bod.

6.1 WDS na Mikrotiku

Pro nastavení WDS a multicastu je nejlepší volbou použití programu Winbox, který je volně ke stažení na stránkách MikroTiku. Po spuštění je nutné zvolit směrovač pomocí IP nebo MAC adresy. Jelikož WinBox nevyžaduje IP adresu, ale sám detekuje připojené zařízení pomocí adresy MAC, není nutné při prvním připojení k RouterBoardu cokoliv měnit, ale postačí síťové spojení pomocí ethernetu nebo sériové linky. Dále je nutné zadat přihlašovací jméno a heslo a připojit se. Program je aktualizován přímo z nainstalovaného RouterOS, tudíž na něj nejsou žádné aktualizace běžně ke stažení. Nastavení lze provést rovněž pomocí konzole, avšak z důvodu přehlednosti je použit WinBox, ve kterém lze konzoli rovněž simulovat.

Po přihlášení do RouterBoardu je v menu *interface* jako první krok nutné povolit bezdrátové rozhraní. V tomto případě je toto rozhraní označeno jako *wlan1*.

Aktivace se provede kliknutím na potvrzovací tlačítko (modré se zatržítkem). Poté se dvojklikem na povolenou kartu otevře dialogové okno s možností konfigurace bezdrátového rozhraní. V záložce *Wireless* musí být nastaven mód na *ap-bridge*. SSID, pásmo a frekvence pak libovolné. Tato nastavení však musí být shodná i na všech ostatních zařízeních, která mají být mezi sebou propojena. Výjimkou může být pouze SSID (popsáno níže). Viz obrázky B.1 a B.2.

Na obrázku B.3 je znázorněn další krok a to přidání mostu, na kterém bude WDS pracovat. Most je vytvořen v menu interface pomocí tlačítka plus a výběru položky most (bridge). V záložce STP (Spanning Tree Protocol) musí být zatržen protokol RSTP (Rapid Spanning Tree Protocol). Ten zajišťuje, aby při komunikaci nedocházelo ke smyčkám a zpětným echům na vlastní zprávy a rovněž detekuje výpadky v síti. Oproti předchozímu protokolu STP je jeho reakce až 10x rychlejší a stačí na ni pouze 3 hello zprávy. Citace dle [8].

Po vytvoření mostu je nutné provést aktivaci WDS ve vlastnostech bezdrátového adaptéru. V záložce WDS je potřeba nastavit mód na dynamic a zadat most, na kterém bude komunikace probíhat. Další možnost je udání módu staticky. Výhodou statického módu je fakt, že při jeho použití mohou mezi sebou komunikovat jen povolená AP. Pro možnost změny SSID na všech AP je nutné zatrhnout „WDS Ignore SSID“. Není-li SSID bráno v úvahu, lze jej použít pro globální roaming s přepojováním klientů. Viz obrázek B.4.

WDS je díky předešlé konfiguraci aktivní. Následujícím krokem (obrázek B.5) je přiřazení virtuálního rozhraní s OpenWRT na stejný most jako WDS. V základním nastavení jsou virtuální rozhraní pojmenována jako vif$X$, kde X označuje číslo v pořadí. První rozhraní bude mít tedy číslo 1 a další hodnoty se inkrementují o jedničku. V menu Bridge → Port, je pomocí tlačítka plus vyvoláno dialogové okno, ve kterém následuje výběr rozhraní (v tomto případě vif$X$) a mostu, ke kterému je přiřazeno rovněž WDS.

Posledním krokem je přiřazení IP adresy fyzickému rozhraní, na které je WDS připojeno. V menu IP → Adresses je potřeba zvolit libovolnou neveřejnou IP adresu z rozsahu typů A, B nebo C. Zbývajícím přístupovým bodům pak nakonfigurovat adresy IP ve stejné podsíti, aby bylo možno uskutečnit spojení. Viz obrázek B.6.

6.2 Multicast

IP multicast je metoda přeposílání IP datagramů z jednoho zdroje skupině více koncových stanic. Místo odesílání jednotlivých datagramů ke každému cíli je odeslán jediný datagram. IP směrování přenosu multicast bylo vyvinuto, aby doplnilo technologie unicast a broadcast, které účinně nezvládaly nové aplikace. Pře-

nášené informace mohou být například v podobě multimédií nebo opravy chyb v operačních systémech apod. Už z vlastnosti přijímání multimediálních dat tedy plyne, že multicast nezaručuje pořadí došlých paketů ani skutečné doručení.

Metoda funguje na principu posílání informací (s IP adresou zdroje a adresou cílové skupiny) ze zdrojového uzlu přes spojení k síti (většinou směrovače) jen jedním datovým tokem, a pokud je informace v lokální síti vyžadována, dojde k její replikaci. Cílem zavedení multicastu je tedy zmenšení zátěže vysílajícího uzlu a přenosové sítě. Citace dle [2].

Pro správné fungování programu je nutné vytvořit přihlašování pro multicastové adresy. Nejdříve je nutné aktualizovat RouterOS na nejnovější dostupnou verzi, kterou povoluje licenční klíč, avšak pro realizaci multicastu postačí verze 4 a vyšší. Ze stránek MikroTiku je v době psaní této práce možno stáhnout RouterOS verze 5.1. Nejjednodušším řešením získání nové verze je výběr a stažení tzv. kombinovaného balíčku, který obsahuje veškeré základní balíčky používané v systému. Postup instalace je následující. Pomocí FTP protokolu (kde je však nutné přenos provádět v binárním módu) nebo pomocí WinBoxu přesunout do menu file celý balíček. Po zkopírování již stačí restartovat směrovač. Aktualizace je provedena automaticky. Dále je však nutné stáhnout i volitelné balíčky, mezi nimiž je obsažen i balíček multicast, bez něhož by v navrženém a realizovaném řešení nemohla komunikace probíhat. Balíček je opět nainstalován stejnou cestou jako aktualizace RouterOS. Po jeho instalaci se objeví v menu rating nová položka a to PIM (*Protocol Independent Multicast*). Dle [2] jde o skupinu směrovacích protokolů, které zařizují distribuci jeden–mnoha nebo mnoho–mnoha přes internet, mezi nimiž je protokol IGMP, který připojuje klienty do multicastových skupin.

V menu Rating → PIM → Interface následuje přidání rozhraní, na kterém WDS pracuje. V tomto případě se jedná o most, ke kterému je WDS přiřazeno. Lze zde rovněž nastavit verzi protokolu IGMP a další možnosti v závislosti na požadavcích. Nicméně pro účely navrženého řešení nejsou další zásahy nezbytné.

7 ZÁVĚR

Seznámení s problematikou sítí v rámci standardu 802.11 a jeho doplňků. Seznámení s operačním systémem RouterOS včetně jeho vlastností a skriptovacího jazyka, který je použit pro vytvoření navrženého programu. Skriptovací jazyk v RouterOS nabízí nepřeberné možnosti v oblasti ovládání a řízení přístupového bodu. Skripty jsou užitečné programy pro administrátory, kterým umožňují automatizovanou správu zařízení bez nutnosti vnějšího zásahu. Lze tedy říci, že v rukou zkušeného programátora představují účinný nástroj pro přeměnu jednoúčelového zařízení na malý multifunkční počítač.

Využití protokolu ICMP jako transportního protokolu vyžaduje dobrou znalost práce se sockety. Sockety operačních systémů obsahují nastavení, která jsou v mnoha případech pro tuto práci nevýznamné. Z tohoto důvodu jsou uvedeny pouze zásadní makra, struktury a nastavení socketu. Nevýhodou socketu typu SOCK_RAW, který je v práci použit, je skutečnost, že jeho vytvoření vyžaduje spuštění aplikace s právy správce. To může mít za následek, v případě chyby ve zdrojovém kódu, nepředvídatelné chování a narušení běhu operačního systému.

Přenos ICMP zpráv probíhá v režimu nespojované služby pomocí datagramů. Nezaručuje tedy spolehlivost jejich doručení. To je výhodou, protože při výpadku jakéhokoliv zařízení v síti nejsou ovlivněny zbývající přístupové body.

Protokolu ICMP může být využito obdobně jako protokolu SNMP, k potřebám správy sítě. Směrovače MikroTik, disponující operačním systémem RouterOS, umožňují správu systému pomocí programového prostředí API. Vytvořenou aplikaci lze dále rozšířit ve formě komunikace s tímto prostředím, které umožní konfiguraci přístupových bodů na základě přenesených informací. Touto cestou je možné provádět automatizovaný dohled sítě.

LITERATURA

[1] IEEE Standards Association *IEEE 802.11*, poslední aktualizace 24. 10. 2010 [cit. 10. 12. 2010]. Dostupné z URL: <http://standards.ieee.org/findstds/standard/802.11-2007.html>.

[2] BOUŠKA, Petr. *Www.samuraj-cz.com* [online]. c2005 [cit. 2011-05-30]. TCP/IP - skupinové vysílání IP Multicast a Cisco. Dostupné z URL: <http://www.samuraj-cz.com/clanek/tcpip-skupinove-vysilani-ip-multicast-a-cisco>.

[3] Mikrotik RouterBOARDS *Mikrotik RB433*, 2. 1. 2008 [cit. 10. 12. 2010]. Dostupné z URL: <http://www.mikrotik.com/testdocs/ros/3.0>.

[4] *Wiki.mikrotik.com* [online]. c2008, 9.5.2011 [cit. 2011-05-30]. Manual:Metarouter. Dostupné z URL: <http://wiki.mikrotik.com/wiki/Manual:Metarouter#Interface>.

[5] *IEEE Std 1003.1-2008* [online]. c2001 [cit. 2011-05-30]. The Open Group Base Specifications Issue 7. Dostupné z URL: <http://pubs.opengroup.org/onlinepubs/9699919799/basedefs/sys_socket.h.html>.

[6] *WDS Linked router network - DD-WRT Wiki* [online]. 14.5.2005, 23.3.2011 [cit. 2011-05-30]. WDS Linked router network. Dostupné z URL: <http://www.dd-wrt.com/wiki/index.php?title=WDS_Linked_router_network>.

[7] *Wds [Marigoldovo WiFi]* [online]. 26.3.2006, 1.12.2008 [cit. 2011-05-30]. Marigoldí WiFi. Dostupné z URL: <http://www.marigold.cz/wifi/doku.php/wds>.

[8] *Understanding Rapid Spanning Tree Protocol (802.1w) [Spanning Tree Protocol] - Cisco Systems* [online]. c2009, 24.10.2006 [cit. 2011-05-30]. Spanning Tree Protocol. Dostupné z URL: <http://www.cisco.com/application/pdf/paws/24062/146.pdf>.

[9] POSTEL, J. *RFC 792 - Internet Control Message Protocol* [online]. 1.9.1981 [cit. 2011-05-30]. Network Working Group. Dostupné z URL: <http://tools.ietf.org/html/rfc792>.

[10] *FreeBSD Developers Handbook* [online]. c2001 [cit. 2011-05-30]. Essential Socket Functions. Dostupné z URL: <http://www.freebsd.org/doc/en_US.ISO8859-1/books/developers-handbook/sockets-essential-functions.html>.

[11] *Mikrotik Wiki* [online]. c2008 [cit. 2011-05-31]. Manual:Scripting. Dostupné z URL: <http://wiki.mikrotik.com/wiki/Manual:Scripting>.

[12] DOSTÁL, Radim. *Root.cz* [online]. 4. 8. 2003 [cit. 2011-06-01]. Sokety a C/C++: program ping. Dostupné z URL: <http://www.root.cz/clanky/sokety-a-c-program-ping/>.

[13] *Linuxdoc.org* [online]. c2001 [cit. 2011-06-01]. Multicast programming. Dostupné z URL: <http://www.linuxdoc.org/HOWTO/Multicast-HOWTO-6.html>.

SEZNAM SYMBOLŮ, VELIČIN A ZKRATEK

Wi-Fi (*Wireless Fidelity*) - česky „bezdrátová věrnost“ je označení pro několik standardů IEEE 802.11, popisujících bezdrátovou komunikaci v počítačových sítích

LAN (*Local Area Network*) - lokální, místní síť, která pokrývá malé geografické území (např. domácnosti nebo malé firmy)

WLAN (*Wireless Local Area Network*) - lokální, bezdrátová, místní síť, která pokrývá malé geografické území (např. domácnosti nebo malé firmy)

MAC (*Medium Access Control*) - je jedinečný identifikátor síťového zařízení, který používají různé protokoly druhé (spojové) vrstvy OSI

ISO (*International Organization for Standardization*) - je světovou federací národních normalizačních organizací se sídlem v Ženevě. Byla založena v roce 1947

EOL (*End of Line*) - sekvence znaků označující konec fyzického řádku. Pro platformu Windows platí *CR* + *LF*, pro Unix *LF* a pro Macintosh *CR*

OFDM (*Orthogonal Frequency Division Multiplexing*) - česky „ortogonální multiplex s kmitočtovým dělením“ je širokopásmová modulace využívající kmitočtové dělení kanálu

MIMO (*Multiple Input Multiple Output*) - česky více vstupů více výstupů, je abstraktní matematický model pro multi-anténní komunikační systémy

SISS (*Support of the Internal Sublayer Service*) - je podklauzule klauzule 2.5, která pokrývá přemosťovací operace v rámci 802.11 MAC podvrstvy

VoWIP (*Voice over Wireless IP*) - je technologie pro vytvoření bezdrátového telefonního systému

QoS (*Quality of Service*) - je v informatice termín používaný pro rezervaci a řízení datových toků v telekomunikačních a počítačových sítích s přepínáním paketů

MIPS (*Microprocessor without Interlocked Pipeline Stages*)

API (*Application Programming Interface*) - označuje rozhraní pro programování aplikací

NEWLINE - znak nebo znaky označující konec fyzického řádku a přechod na řádek nový

CR (*Carriage Return*) - česky „návrat vozíku“ je netisknutelný formátovací znak, který přesouvá kurzor na začátek řádku

LF (*Line Feed*) - česky „posun o řádek“ je netisknutelný formátovací znak, který přesouvá kurzor na další řádek

FF (*Form Feed*) - česky „posun o stránku“ je netisknutelný formátovací znak, který přesouvá kurzor na další stránku

HT (*Horizontal Tabulation*) - česky „vodorovná tabulace“ je netisknutelný formátovací znak, který přesouvá kurzor na další znak téhož řádku

VT (*Vertical Tabulation*) - česky „svislá tabulace“ je netisknutelný formátovací znak, který přesouvá kurzor na stejnou pozici dalšího řádku

BEL (*Bell*) - česky „zvonek“ je netisknutelný kontrolní znak, který označuje místo výstrahy

RouterOS (*Router Operation System*) - je operační systém používaný na směrovačích firmy MikroTik

FTP (*File Transfer Protocol*) - je protokolem pro přenos souborů

SFTP (*Secure File Transfer Protocol*) - je zabezpečeným protokolem pro přenos souborů

MTU (*Maximum Transmission Unit*)

PF (*Protocol Family*)

AF (*Address Family*)

TTL (*Time To Live*)

PIM (*Protocol Independent Multicast*)

ICMP (*Internet Control Message Protocol*)

WDS (*Wireless Distribution System*)

SNMP (*Simple Network Management Protocol*)

SEZNAM PŘÍLOH

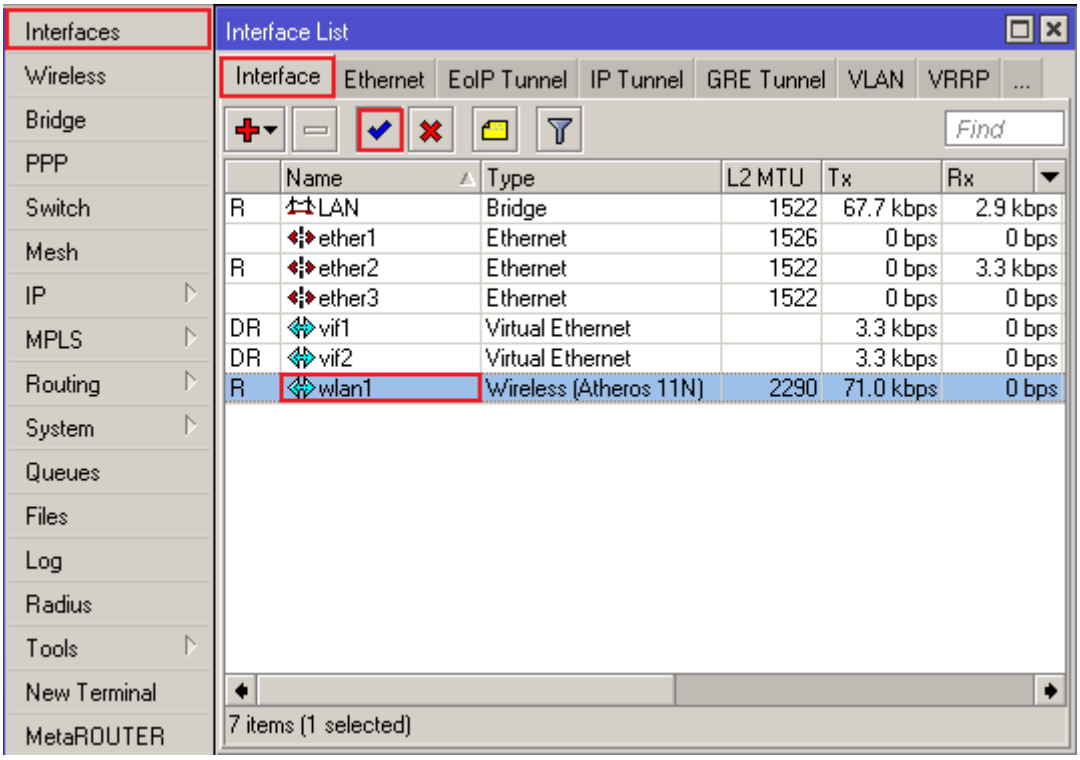
A TABULKY

Tab. A.1: Globální příkazy v RouterOS

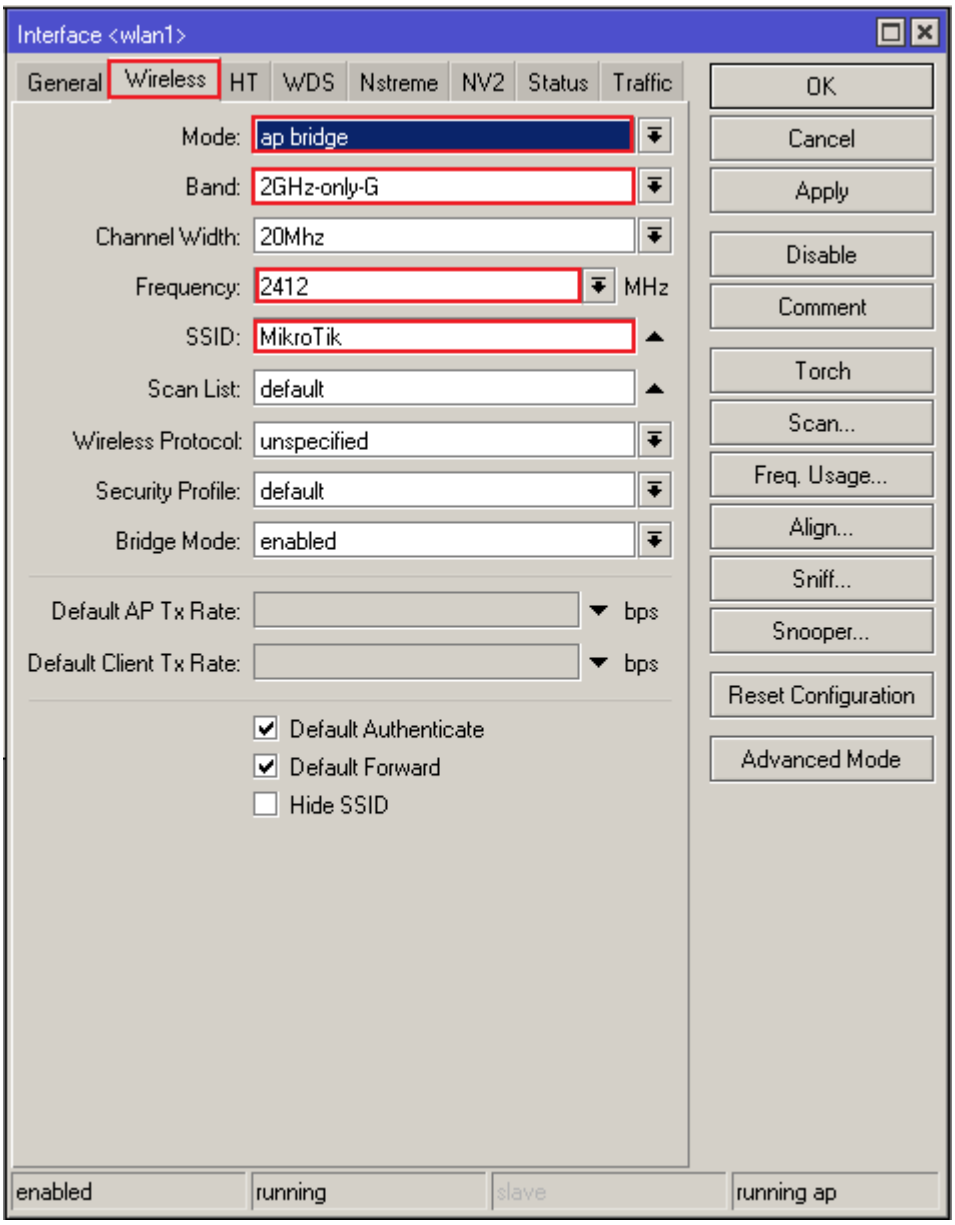
Příkaz	Syntaxe	Význam
/		přesun do kořenového adresáře menu
..		přesun v menu zpět o jednu úroveň
?		výpis nápovědy
global	:global <proměnná> [<hodnota>]	deklarace globální proměnné
local	:local <proměnná> [<hodnota>]	deklarace lokální proměnné
beep	:beep <frekvence> <délka>	pípnutí vestavěného reproduktoru
delay	:delay <čas>	pomlka na daný časový úsek v sekundách
put	:put <výraz>	vytiskne do konzole zpracovaný výraz
len	:len <výraz>	vrátí délku řetězce nebo počet prvků pole
typeof	:typeof <proměnná>	vrátí datový typ proměnné
pick	:pick <proměnná> <start> [<konec>]	vrátí rozsah prvků nebo dílčího řetězce. Pokud není definován konec, vrátí pouze jeden prvek pole
log	:log <téma> <zpráva>	zapíše zprávu do systémového logu. Povolená témata jsou *debug*, *error*, *info* a *warning*
time	:time <výraz>	vrátí časový interval potřebný pro zpracování výrazu
set	:set <proměnná> [<hodnota>]	přiřadí hodnotu deklarované proměnné
find	:find <argument> <argument> <start>	vrátí pozici podřetězce v řetězci nebo prvku v poli
environment	:environment print [<start>]	vytiskne informace o inicializované proměnné
terminal		terminálové související příkazy
error	:error <výstup>	generuje chybu a zastaví běh programu
parse	:parse <výraz>	zpracuje výraz a vrátí zpracované příkazy. Může být použit jako funkce
resolve	:resolve <argument>	vrátí IP adresu zadané domény

toarray	:toarray <proměnná>	konverze proměnné na pole
toarray	:toarray <proměnná>	konverze proměnné na pole
tobool	:tobool <proměnná>	konverze proměnné na pravda - nepravda
toid	:toid <proměnná>	konverze proměnné na interní číslo
toip	:toip <proměnná>	konverze proměnné na IP adresu verze 4
toip6	:toip6 <proměnná>	konverze proměnné na IP adresu verze 6
tonum	:tonum <proměnná>	konverze proměnné na číslo
tostr	:tostr <proměnná>	konverze proměnné na řetězec znaků
totime	:totime <proměnná>	konverze proměnné na datum a čas

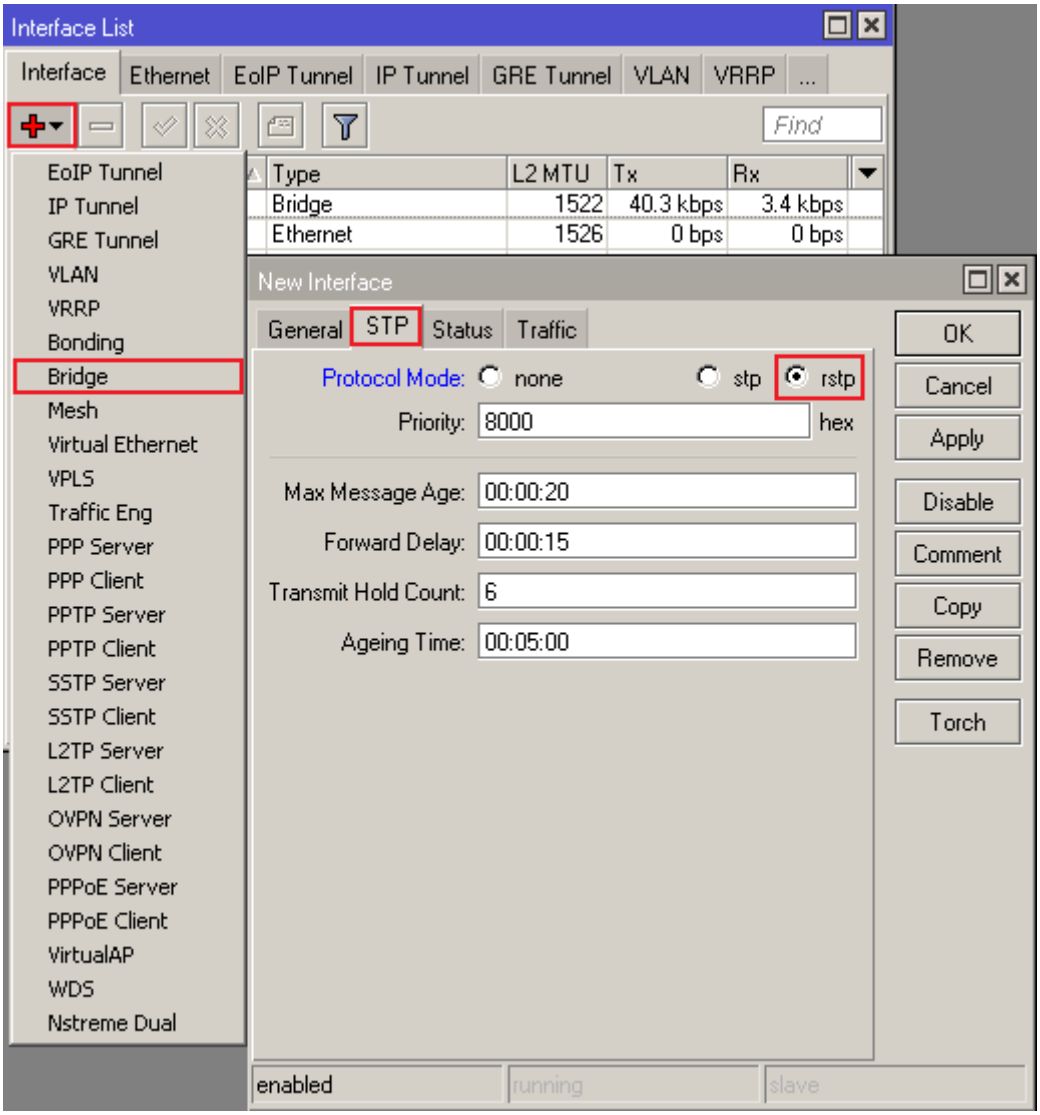
B OBRÁZKY



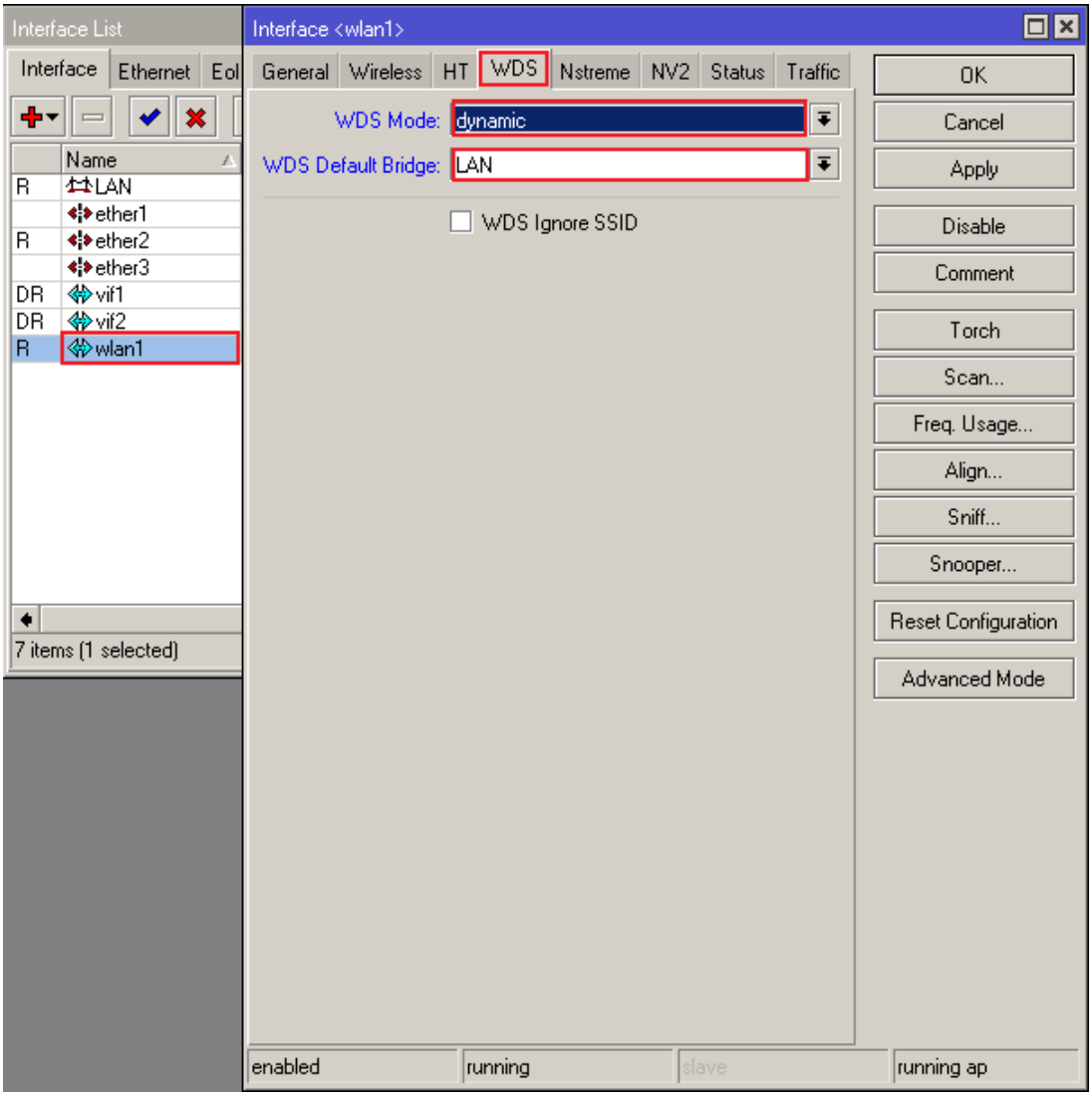
Obr. B.1: Aktivace bezdrátového rozhraní



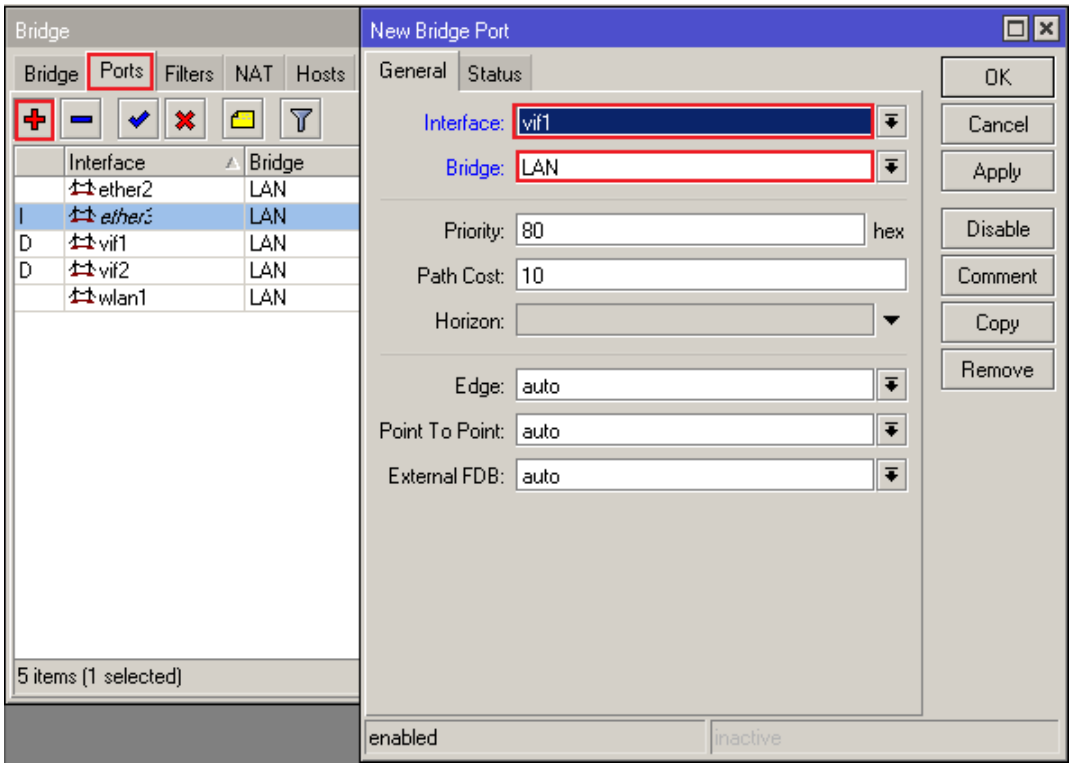
Obr. B.2: Nastavení bezdrátového rozhraní



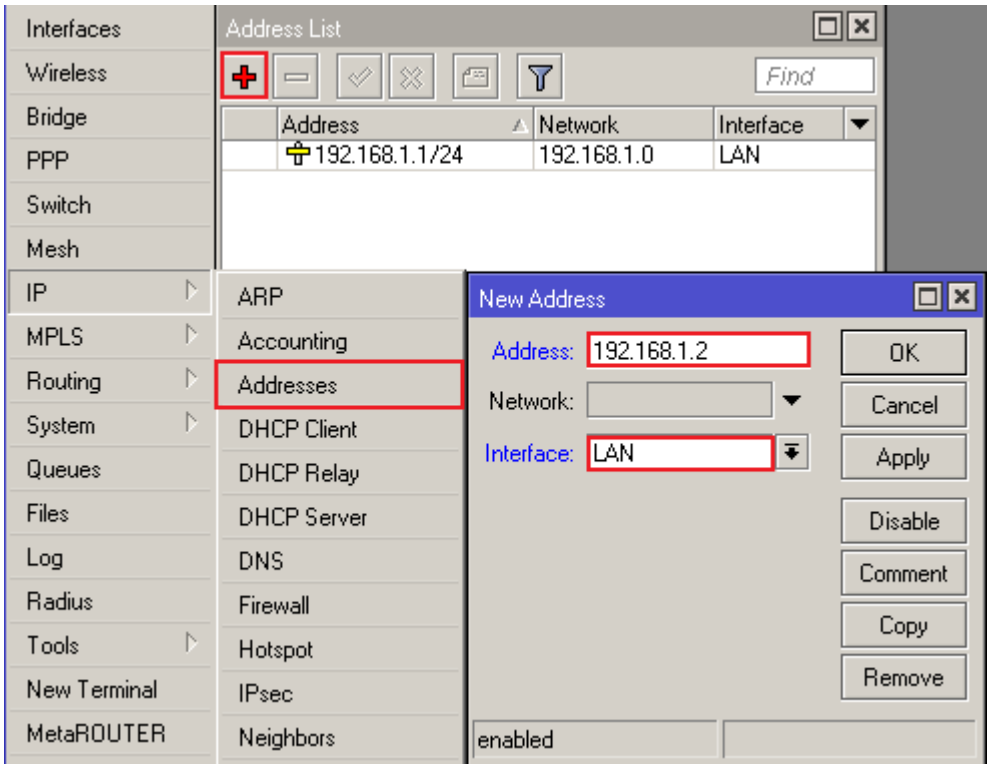
Obr. B.3: Nastavení protokolu RSTP



Obr. B.4: Nastavení WDS



Obr. B.5: Přiřazení virtuálního rozhraní



Obr. B.6: Přiřazení adresy IP

```
#include <netinet/in.h>
#include <sys/socket.h>
#include <sys/types.h>
#include <netinet/ip_icmp.h>
#include <unistd.h>
#include <stdio.h>

#define BUFSIZE 1024

/* Definice proměnných */
int sock;
char buffer[BUFSIZE];
int bufferLenght;
char loopch = 0;
icmpdata *ic;
sockaddr_in sendSockAddr;
fd_set mySet;
timeval tv;
ip_mreq mreq;
in_addr address;

/* Vytvoření socketu */
sock = socket(PF_INET, SOCK_RAW, IPPROTO_ICMP);

/* Nastavení skupinového vysílání */
mreq.imr_multiaddr.s_addr = inet_aton("239.0.0.1");
address = mreq.imr_interface.s_addr = inet_aton("192.168.1.2");

/* Nastavení voleb socketu */
setsockopt(sock, IPPROTO_IP, IP_MULTICAST_IF, &address , sizeof(in_addr));
setsockopt(sock, IPPROTO_IP, IP_MULTICAST_LOOP, &loopch, sizeof(char));
setsockopt(sock, IPPROTO_IP, IP_ADD_MEMBERSHIP, &mreq, sizeof(ip_mreq));

/* Alokace paměti pro ICMP zprávu s odesílanými data */
ic = (icmpdata *)malloc(sizeof(icmpdata));
```

Obr. B.7: Ukázka použití funkcí a knihoven č. 1

```
/* Nastavení atributů ICMP hlavičky */
/* Typ a kód ICMP zprávy lze volit libovolně */
ic->icmp.type = ICMP_IREQ;
ic->icmp.code = 0;
/* Funkce checksum opsána z příslušných RFC dokumentů */
ic->icmp.checksum = checksum((const short unsigned int*)&ic, sizeof(icmpdata));
/* Data jsou prozatím naplněna konstantními daty */
ic->used = 92;
ic->count = 1000;

/* Příprava odesílání dat */
sendSockAddr.sin_family = AF_INET;
sendSockAddr.sin_port = 0;
sendSockAddr.sin_addr = address;

/* Odeslání dat */
sendto(sock, (char *)ic, sizeof(icmpdata), 0, (sockaddr *)&sendSockAddr,
       sizeof(sockaddr_in));

/* Nastavení časovače pro funkci select */
tv.tv_sec = 20;
tv.tv_usec = 0;

/* Vyprázdnění a naplnění množiny fd_set vytvořeným socketem */
FD_ZERO(&mySet);
FD_SET(sock, &mySet);

/* Použití funkce select čekající událost na socketu */
select(sock + 1, &mySet, NULL, NULL, &tv);

/* Podmínka ověření naplnění množiny a příjmutí zprávy */
if (FD_ISSET(sock, &mySet))
{
bufferLenght = sizeof(sockaddr_in);
recvfrom(sock, buffer, BUFSIZE, 0, (sockaddr *)&sendSockAddr, &bufferLenght)
}
```

Obr. B.8: Ukázka použití funkcí a knihoven č. 2

C PROGRAM ICMPMULTICAST

Program je spouštěn s parametry. Veškeré adresy jsou zadávány v tečkovém formátu.

- [**MULTICAST_ADDRESS**] – použitá skupinová adresa
- [**ROUTER_IP**] – IP adresa brány (mostu) fyzického zařízení, ke kterému je připojen virtuální systém
- [**USERNAME**] – přihlašovací jméno do RouterOS
- [**PASSWORD**] – heslo do RouterOS
- [**WRT_IP**] – IP adresa použitého rozhraní v OpenWRT
- [**INTERVAL**] – interval odesílání zpráv

Parametry jsou povinné a jejich pořadí musí být striktně dodrženo. Přijaté zprávy jsou vypisovány do konzole.