FAKULTA INFORMAČNÍCH TECHNOLOGIÍ ČVUT V PRAZE

Zadání bakalářské práce

Název:	Emulace rozhraní Cisco směrovačů v operačním systému OpenWRT
Student:	Filip Dovalil
Vedoucí:	Ing. Viktor Černý
Studijní program:	Informatika
Obor / specializace:	Počítačové sítě a Internet 2021
Katedra:	Katedra počítačových systémů
Platnost zadání:	do konce letního semestru 2025/2026

Pokyny pro vypracování

Vytvořte aplikaci pro operační systém OpenWRT, která emuluje chování rozhraní Cisco IOS. Aplikaci bude možno používat jako login shell pro uživatele přihlášené přes SSH a umožní konfiguraci síťového prvku stejně jako na Cisco IOS. Funkčnost výsledného produktu demonstrujte na příkazech, které pokrývají řešení laboratorních úloh v předmětu Počítačové sítě (BI-PSI).

Elektronicky schválil/a prof. Ing. Pavel Tvrdík, CSc. dne 29. února 2024 v Praze.

Bakalářská práce

EMULACE ROZHRANÍ CISCO SMĚROVAČŮ V OPERAČNÍM SYSTÉMU OPENWRT

Filip Dovalil

Fakulta informačních technologií
Katedra počítačových systémů
Vedoucí: Ing. Viktor Černý
16. května 2024

Obsah

Seznam obrázků

Tímto bych chtěl poděkovat především svému vedoucímu práce panu Ing. Viktorovi Černému za pomoc při realizaci této bakalářské práce, potřebné konzultace a veškerý věnovaný čas. Poděkování patří také mé rodině za nezbytnou podporu nejen při psaní této práce ale i po celou dobu studia.

Prohlášení

Prohlašuji, že jsem předloženou práci vypracoval samostatně a že jsem uvedl veškeré použité informační zdroje v souladu s Metodickým pokynem o dodržování etických principů při přípravě vysokoškolských závěrečných prací.

Beru na vědomí, že se na moji práci vztahují práva a povinnosti vyplývající ze zákona č. 121/2000 Sb., autorského zákona, ve znění pozdějších předpisů, zejména skutečnost, že České vysoké učení technické v Praze má právo na uzavření licenční smlouvy o užití této práce jako školního díla podle § 60 odst. 1 citovaného zákona.

V Praze dne 16. května 2024

Abstrakt

Tato bakalářská práce se zabývá návrhem a implementací programu, který emuluje rozhraní systému Cisco IOS v operačním systému OpenWRT. V teoretické části práce jsou analyzovány vlastnosti a funkce, které jednotlivé operační systémy nabízejí, poté jsou rozebírány jednotlivé použité technologie pro emulaci. Praktická část se zaměřuje na konkrétní postupy při implementaci jednotlivých funkcí emulačního prostředí. Následně jsou rozebírána zjištěná omezení a limity dostupných nástrojů v rámci systému OpenWRT a jejich dopad na výslednou funkčnost emulace. V závěru práce jsou navrženy další možné kroky vedoucí k rozvoji a zpřesnění navrženého emulačního rozhraní a jsou zde shrnuty možnosti použití a zhodnocení životaschopnosti tohoto projektu.

Klíčová slova emulace Cisco IOS, OpenWRT, jazyk C, skriptování v shellu, GNS3, síťová komunikace

Abstract

This bachelor's thesis deals with the design and implementation of a program that emulates the interface of the Cisco IOS system in the OpenWRT operating system. The theoretical part of the thesis analyzes the characteristics and functions offered by individual operating systems, followed by the discussion of the various technologies used for emulation. The practical part focuses on specific procedures for implementing the various functions of the emulation environment. In addition the identified limitations and constraints of the available tools within the OpenWRT system are analyzed, along with their impact on the resulting functionality of the emulation. In conclusion, further possible steps leading to the development and refinement of the proposed emulation interface are suggested, and the possibilities for use and evaluation of the viability of this project are summarized.

Keywords Emulation of Cisco IOS, OpenWRT, C language, shell scripting, GNS3, network communication

Seznam zkratek

ACL	access control list
Cisco IOS	Cisco Internetworking operating system
DHCP	dynamic host configuration protocol
GNU	GNU not unix
GUI	graphical user interface
IP	internet protocol
LAN	local area network
NAT	network address translation
OSI	open systems interconnection
OSPF	open shortest path first
PAT	port address translation
POSIX	portable operating system interface
RAM	random access memory
SDK	software development kit
SSH	secure shell
UCI	unified configuration interface
WAN	wide-area network

Úvod

Správně nastavená síťová komunikace je základním stavebním kamenem pro korektně fungující propojení jednotlivých komponent v počítačových sítích. Směrovač a jeho bezchybná konfigurace jsou tak klíčovými prvky ve vzájemném propojování počítačových segmentů. Prvotním úkolem tohoto prvku je správné směrování dat mezi různými počítačovými sítěmi. Bez směrovačů by bylo velmi obtížné, pokud ne nemožné, sdílet informace a data mezi spojenými sítěmi.

Zařízení pracují na třetí vrstvě modelu OSI, známé také jako síťová vrstva. Tato vrstva se zabývá adresací a směrováním datových paketů a úkolem směrovače je přijímat příchozí pakety s daty, analyzovat jejich cílové adresy a poté je nasměrovat do správného výstupního portu směrem k jejich cílové destinaci. Směrovače také hrají klíčovou roli v zabezpečení sítě. Mnoho směrovačů má vestavěné funkce pro ochranu sítě, jako je například firewall, který blokuje nežádoucí provoz, nebo implementuje nástroje pro šifrování dat, které pomáhají chránit přenášené citlivé informace.

Společnost Cisco Systems je v oblasti síťové komunikace lídrem. Definuje nejen postupy, ale vytváří i celosvětové standardy na poli síťových technologií. Možnost využití daných postupů a rozšíření prostředí Cisco i do dalších zařízení, umožní ovládat a nastavovat tyto prvky bez nutnosti náročného studia dalších odlišných konfiguračních prostředí jiných výrobců.

Nabízí se tak možnost napodobit chování jednoho systému na systému jiném, tedy provést emulaci. To s sebou přináší řadu výzev a problémů. Jedním z hlavních problémů je výkon, protože emulace může být náročná na procesor a paměť, což může snižovat rychlost výměny informací mezi komponentami hostitelského systému. Dalším problémem při tvorbě prostředí, které napodobujeme, je kompatibilita. Ne všechny funkce nebo schopnosti hostovaného systému mohou být implementovány na hostitelském systému, což může přinášet potíže s funkcionalitou i stabilitou celého software. Emulace také představuje výzvy v oblasti zabezpečení. Vzniklé nové prostředí může být více zranitelné a méně odolné proti různým druhům útoků. Navíc, pokud je emulovaný systém kompromitován, případné následky mohou mít dopad i na hostitelský systém. I přes výše uvedené problémy přináší emulace mnoho výhod. Tou hlavní z po-

hledu této práce je nabídnutí prostředí Cisco širší veřejnosti a zjednodušení konfigurace neznačkových zařízení odborníkům v oblasti Cisco zařízení.

Platformou, kde je možné emulovat chování Cisco prostředí, je dle zadání OpenWRT. Tato platforma je, na rozdíl od komerčních Cisco systémů, volně šiřitelná. Z dostupných zdrojů není známo, že by se někdo o tento typ emulace Cisco IOS v OpenWRT úspěšně pokusil.

Kapitola 1

Cíle práce

Cílem této bakalářské práce bude ověřit, zda lze emulovat prostředí směrovačů Cisco v rámci open-source systému OpenWRT. Prostředí Cisco IOS disponuje komplexní množinou příkazů pro management veškerého síťového provozu nejen v lokálních ale i rozsáhlých počítačových sítích. Úkolem práce však bude implementovat pouze základní konfigurační příkazy ze směrovačů postavených na tomto komerčním produktu, do prostředí OpenWRT tak, aby bylo možné řídit síťový provoz na protokolu IP verze 4 v jednoduché počítačové síti a ověřit tak funkčnost navrženého řešení. Množina emulovaných příkazů odpovídá rozsahu předmětu BI-PSI, vyučovaném na FIT ČVUT. Řízení a ovládání směrovače bude prováděno prostřednictvím SSH spojení, nastavení konfigurace nově vytvořeného prostředí pak bude probíhat prostřednictvím terminálového módu. Emulace grafického prostřední není předmětem práce. Základní platformou pro testování bude zvoleno virtualizační prostředí GNS3, nabízející všechny nutné podmínky pro vytvoření odpovídajícího technického zázemí dovolujícího provoz, jak virtuálních směrovačů Cisco nebo OpenWRT, tak nasazení monitorovacích nástrojů.

Ze získaných zkušeností při vývoji a testování emulačního prostředí na platformě OpenWRT bude možné vyhodnotit životaschopnost celého projektu a posoudit možnosti dalšího rozvoje vytvořeného prostředí. Samotné zkušenosti nabyté při tvorbě aplikace také stanoví limity převodu prostředí Cisco do volně šiřitelného projektu a nastíní cestu k dalšímu rozšiřování množiny dostupných příkazů. Výstupem celé bakalářské práce bude souhrn doporučení, za jakých podmínek, a s jakými omezeními lze dále rozvíjet emulaci systému Cisco IOS v jiném prostředí, konkrétně na platformě OpenWRT.

Kapitola 2
Analýza

Následující kapitola se zabývá základním, obecným popisem operačních systémů Cisco IOS a OpenWRT, jejich vlastností a funkčností. Dále jsou rozebírány jednotlivé technologie, které byly brány v potaz při výběru nástrojů pro konstrukci emulátoru.

Za vzorové prostředí a chování Cisco IOS je považováno zařízení Cisco 3725, v rámci kterého je použita verze Cisco IOS s označením 12.4(15)T7. V důsledku nedostupnosti fyzického směrovače Cisco při vytváření emulace je image tohoto zařízení provozován v rámci programu GNS3.

Vzhledem k uzavřenosti operačního systému Cisco IOS není možné během vývoje nahlížet přímo do zdrojových kódů a zkoumat, jak jsou dané funkce implementovány. Z tohoto důvodu jsou reakce systému na určité testované vstupy brány jako závazné a vývoj emulace je vytvářen na základě konkrétní odezvy systémů na dané konfigurační podněty.

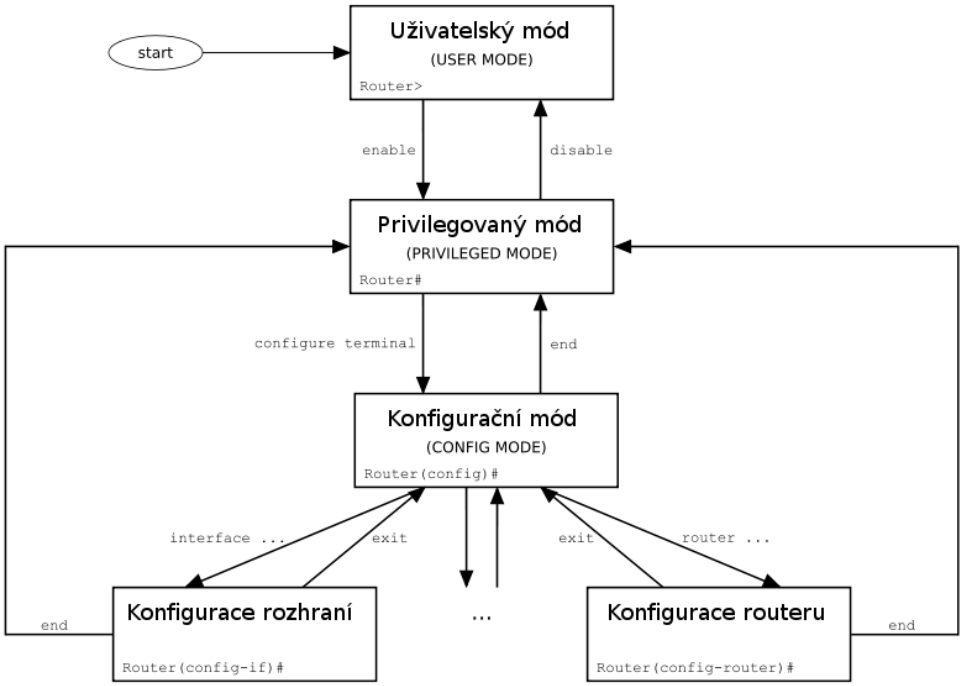
2.1 Cisco IOS

Cisco IOS je komplexní operační systém vyvinutý společností Cisco Systems určený pro směrovače, přepínače a další síťová zařízení dodávané touto společností. Jedná se o monolitický operační systém, vyvíjený převážně v jazyce C, s upraveným jádrem. Novější verze tohoto systému jsou již založené na klasickém linuxovém jádře. Vedle standardního Cisco IOS existují ještě tři další varianty tohoto operačního systému, a to: IOS XE, IOS XR, NX-OS. Každá ze zmíněných variant je navržena a upravena pro jiné potřeby konkrétního síťového provozu od malých řešení až po cloudové služby.[1]

2.1.1 Popis prostředí

Pro ovládání a konfiguraci daného zařízení poskytuje Cisco IOS vlastní příkazové rozhraní. Nadstavbou k této variantě konfigurace přístroje je i zabezpe-

čené webové rozhraní, které pomocí precizně zpracovaného GUI dává obsluze srozumitelnější přehled o jeho stavu. Pro zabezpečení a strukturalizaci celého systému rozděluje Cisco IOS konfiguraci síťového prvku do několika konfiguračních módů. Každý z těchto režimů poskytuje uživateli přístup k jiné části konfigurace ovládaného zařízení. Téměř všechny tyto módy obsahují vlastní sadu příkazů pro monitorování a konfiguraci síťového prvku. Obrázek 2.1 znázorňuje základní koncept a hierarchii zmíněných režimů, které jsou, v rámci operačního systému Cisco IOS, dostupné.[2]



Obrázek 2.1 Operační módy Cisco IOS [3]

Po úspěšném přihlášení na dané zařízení je uživateli poskytnut výchozí neprivilegovaný, uživatelský režim. V rámci tohoto módu je uživateli zpřístupněna pouze omezená sada příkazů, která tvoří podmnožinu příkazů, dostupných v dalším režimu a není umožněna žádná konfigurace zařízení.

Přechodem do privilegovaného režimu je uživatel oprávněn zobrazit veškerou aktivní konfiguraci a statistiky ovládaného zařízení. Tento režim dokáže již uživateli poskytnout citlivé informace o běhu a konfiguraci daného přístroje, a proto je doporučeno chránit přechod z uživatelského módu komplexním administrátorským heslem. V následujícím globálním konfiguračním módu uživatel upravuje konfiguraci, která má dopad na chování zařízení jako celku. Tento režim slouží také jako výchozí pro přechod do dalších konfiguračních režimů, které již umožní upravovat konfiguraci přímo vybraných sekcí, jako například nastavení vybraného síťového rozhraní či směrovacího protokolu.

V rozsahu celého příkazového rozhraní podporuje Cisco IOS používání příkazových zkratek, je-li každý ze zadaných argumentů příkazového řádku pro systém jednoznačně interpretovatelný. Pokud pro uživatele není použití zkratek vhodné, může využitím tabulátoru danou jednoznačně identifikovatelnou zkratku doplnit do podoby celého argumentu. Zapomene-li nebo nezná-li uživatel syntaxi daného příkazu, nabízí Cisco IOS možnost pomocí příkazu *?* zobrazit všechny následující možnosti, které mohou být v rámci kontextu již zadaného příkazu použity. Veškeré provedené konfigurační změny jsou aplikovány pouze na běžící konfiguraci, uložené v paměti RAM, a bez zavolání explicitních příkazů pro uložení do perzistentní paměti, je změněná konfigurace při restartu zařízení bez možnosti obnovení ztracena.

Potřebuje-li uživatel průběžně zobrazovat provedené změny v konfiguraci bez nutnosti opakovaně přecházet do režimu obsahujícího sadu příkazů pro tyto účely, implementuje systém pro tyto situace příkaz *do*, umožňující volání téměř všech příkazů, které jsou běžně dostupné v privilegovaném režimu, i v rámci jakéhokoliv konfiguračního módu.

Pokud uživatel provede špatnou konfiguraci, je možné příkazy z používané konfigurace jednoduše odebrat vložením celého příkazu do příkazové řádky za klíčové slovo *no*.

2.2 OpenWRT

OpenWRT představuje kompletní open-source operační systém, který je vystavěn na OS Linux. Jeho použití je cíleno na vestavěné systémy, přednostně pak na směrovače a další zařízení, která pracují se síťovou komunikací. Samotný projekt OpenWRT je vyvíjen již přes 20 let a i nadále zůstává ve fázi aktivního vývoje a vylepšování. Ačkoliv se nejedná o systém, primárně zaměřený na profesionální zařízení, která jsou používána v rámci velký firemních či páteřních sítích, jako tomu je u Cisco IOS, může systém OpenWRT představovat jednu z možných alternativ pro nahrazení původního firmwaru, který do daného zařízení nahrál výrobce. Výrobcem dodané předinstalované programy a nabízené vlastnosti totiž nemusí splňovat požadavky uživatele nutné pro kompletní přizpůsobení správy síťového provozu. Na rozdíl od těchto případů OpenWRT poskytuje flexibilní a transparentní systém, který uživateli nabízí kompletní kontrolu nad daným zařízením a umožňuje modifikaci služeb, systémových parametrů a v případě nutnosti i přepsání komponent samotného jádra.[4]

2.2.1 Možnosti modifikace systému

Výchozí distribuce OpenWRT, na rozdíl od Cisco IOS, poskytuje uživateli možnost pouze základní konfigurace daného zařízení. Úprava a rozšíření možností běžícího zařízení, přidávání dalších funkcionalit a celkové přizpůsobení systému specifickým potřebám je prováděno prostřednictvím instalačních balíčků

z ověřených repozitářů, přes systémového správce balíčků opkg. V případě, že je daná funkcionalita v balíčku pro potřeby uživatele implementovaná nevhodně nebo ji žádný balíček neobsahuje, poskytuje OpenWRT možnost vytvoření vlastního aplikačního balíčku. Tento proces uživatelských úprav může být proveden buď pomocí vývojového prostředí (SDK) nebo vytvořením nového firmware image.

Vývojové prostředí SDK obsahuje již všechny potřebné zkompilované komponenty, nutné k vytvoření nových balíčku, a nabízí tak programátorovi ucelené prostředí pro další vývoj systému. Tyto nástroje jsou aplikovány v různých částech kompilace pro určenou cílovou platformu a to bez nutnosti rekompilace celého systému od samotného základu. Výstupem tohoto tvůrčího procesu je poté soubor s příponou .ipk, reprezentující vytvořený balíček, který je možné prostřednictvím správce balíčků rozbalit a instalovat přímo do vybraného systému OpenWRT.[5]

Pro kompilaci a vytvoření vlastního OpenWRT image je nutné nejprve nastavit vhodné vývojového prostředí. To vyžaduje použití nativního nebo virtualizovaného GNU/Linux prostředí s case-sensitive souborovým systémem. Následující příkaz slouží pro instalaci všech nástrojů, tvořící požadované prerekvizity pro použití kompilačního systému OpenWRT, do vývojového prostředí (uvedeno pro linuxovou distribuci Ubuntu). [6]

- *sudo apt install build-essential clang flex bison g++ gawk gcc-multilib g++-multilib gettext git libncurses5-dev libssl-dev python3-setuptools rsync swig unzip zlib1g-dev file wget* [6]

Žádná jiná prostředí než unixová, nejsou zatím pro využití kompilačního procesu oficiálně podporována.

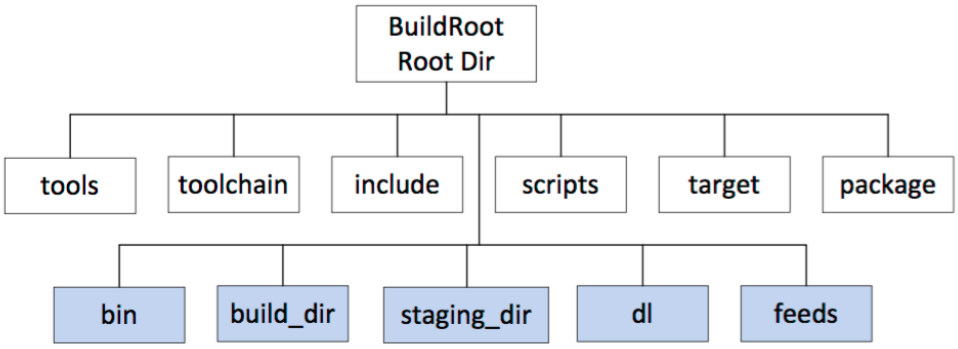
Samotný výsledný image, tedy celkový funkční obraz směrovače, je možné zkompilovat nástrojem ImageBuilder nebo provést kompilaci z volně stažitelných zdrojových kódů OpenWRT z gitlab repozitáře.

Dostupný nástroj ImageBuilder stahuje již předkompilované systémové balíčky, namísto jejich kompletní kompilace přímo ze zdrojových kódů. Tento krok sice zrychluje a zjednodušuje popsaný kompilační proces, ale vzhledem k tomu, že jsou balíčky již předkompilovány, některé možnosti přizpůsobení výsledného image nelze provést. Příkladem muže být výběr cílové platformy pro kompilaci, která je předem určena při stahování nebo znemožnění úprav chování a funkcionalit zahrnutých balíčků.[7]

Naopak při využití repozitáře se zdrojovými kódy je celý systém kompilace řízen nástrojem buildroot, který podobně jako na OS Linux dovoluje proces kompilace, bez jakýchkoli omezení, přizpůsobit potřebám vývojáře. Kompilace se skládá z několika Makefile souborů a patchů, které se starají o korektní průběh sestavení a automatizují proces vytvoření křížového kompilátoru (cross-compiler) pro zvolenou platformu. Takto vytvořený scénář následně slouží pro kompilaci samotného linuxového jádra, root filesystému a dalších instalačních

balíčků, potřebných pro spuštění OpenWRT na zvolené platformě, která může zpracovávat odlišnou instrukční sadu od platformy vývojového prostředí.[8]

Adresář, do kterého se daný repozitář stáhne, je zároveň také root adresářem celého kompilačního procesu. Na obrázku 2.2 je znázorněna stručná adresářová struktura staženého repozitáře. Modře zvýrazněné adresáře jsou generovány během kompilace.



Obrázek 2.2 Adresářová struktura nástroje Buildroot [9]

Popis důležitých adresářů obrázku 2.2:

- bin – adresář, do kterého jsou kompilovány jednotlivé balíčky a také výsledný firmware image
- tools – adresář, který obsahuje nástroje a instrukce potřebné k vytvoření toolchainu, ke kompilace balíčků a vytvoření image
- toolchain – adresář, obsahuje veškeré instrukce a nástroje, včetně standardní C knihovny nebo kompilátoru, zodpovědné za samotné vytvoření výsledného firmwaru
- package – adresář sdružující kompilační instrukce jednotlivých balíčků a jejich zdrojové kódy
- target – adresář, obsahující nástroje specifické pro cílovou platformu, pro kterou je výsledný image určen [9]

Samotný proces vytváření image je poměrně náročná operace pro hardwarové prostředky vývojového prostředí. Pro image, které obsahují pouze základní předvybrané balíčky, je třeba okolo 10 až 15 GB volné místa na disku pro potřeby všech nástrojů a souborů, použitých při sestavení image. Samotná fáze kompilace poté vyžaduje dle oficiální dokumentace 2 až 4 GB paměti RAM v závislosti na zvolené cílové platformě.[10]

Pro účely této práce je použit vývoj prostřednictvím vytvoření vlastního aplikačního balíčku pro OpenWRT, který je kompilován v rámci procesu vytváření upravené OpenWRT image. Hlavními důvody pro tento přístup bylo

poskytnutí maximální možnosti přizpůsobení systému kladeným požadavkům včetně možnosti výběru jednotlivých systémových balíčků ještě před samotnou kompilací. Možnost upravení cílové platformy v případě, že by byl image instalován na více různých zařízení, je v tomto případě také přínosem toho způsobu vývoje. Konkurenční nástroj ImageBuilder neumožňuje kompilaci vlastních vytvořených balíčků přímo ze zdrojových kódů, ale poskytuje pouze možnost naimportovat již hotové balíčky do vytvářené image, což představuje ve vývoji zásadní omezení. V neposlední řadě byl brán ohled i na možný budoucí rozvoj tohoto projektu, který by mohl požadovat větší zásahy do struktur OpenWRT.

2.2.2 Dostupné nástroje v OpenWRT

Systém OpenWRT se snaží zůstat co nejvíce minimalistický, aby byl umožněn jeho běh i na zařízeních s velmi omezenými prostředky. Tomuto požadavku jsou uzpůsobeny také veškeré výchozí dostupné nastavení systému. Interakci s uživatelem pomocí příkazového řádku v systému OpenWRT zajišťuje jednoduchý příkazový interpret ash. Jedná se o malý, POSIX kompatibilní shell, který je populární i v rámci vestavěných systémů. V prostředí OpenWRT je ash součástí výchozího používaného nástroje Busybox, který sdružuje základní, dostupné unixové nástroje v rámci jednoho binárně spustitelném souboru. Zahrnuté komponenty obecně neposkytují všechnu dostupnou funkcionalitu jako standardní prostředky GNU, avšak tento kompromis dovoluje ušetřit nemalé paměťové prostředky daného zařízení. Cílem nástroje Busybox je zajištění kompletního provozního prostředí i pro zařízení s malým výkonem a v případě potřeby je možné funkcionality zahrnutých nástrojů v rámci Busybox upravovat před samotnou kompilací výsledného image.[11]

2.2.2.1 Rozhraní UCI

UCI představuje hlavní konfigurační rozhraní v systému OpenWRT a je využíváno pro efektivní analýzu a nastavování klíčových vlastností daného prostředí. Přehlednost UCI spočívá v organizaci konfiguračních nastavení do hierarchické stromové struktury, což umožňuje snadnou navigaci a správu. Využívají se textové konfigurační soubory uložené v adresáři /etc/config/, což umožňuje jak ruční úpravy, tak i programové manipulace s nastavováním chování systému. Struktura těchto souborů je přizpůsobena tak, aby byla zajištěna snadná editace nástrojem UCI nebo pomocí vestavěného webového rozhraní nesoucí název Luci. Toto grafické rozhraní však není vzhledem k charakteru zadání práce používáno.[12]

Konfiguraci pomocí UCI často podporují také dostupné instalační balíčky, což přispívá k přehlednému a jednotnému systému ovlivňování chování celého zařízení. Nástroj UCI tak zjednodušuje proces konfigurace OpenWRT a umožňuje uživatelům přehlednou interakci se systémovou konfigurací řízeného zařízení.

Obrázek 2.3 poskytuje náhled základní struktury jednoho z výchozích konfiguračních souborů /etc/config/network, který ovládá nastavení jednotlivých dostupných rozhraní. Obrázek demonstruje nastavení síťových rozhraní, které je používáno v prostředí emulátoru.

```
config interface 'FastEthernet00'
        option device 'eth0'
        option proto 'static'
        option ipaddr '192.168.0.1'
        option netmask '255.255.0.0'

config interface 'FastEthernet01'
        option device 'eth1'
        option proto 'static'
        option ipaddr '1.0.0.1'
        option netmask '255.0.0.0'

config interface 'FastEthernet02'
        option device 'eth2'
        option proto 'dhcp'
```

Obrázek 2.3 Příklad syntaxe konfiguračních souborů používajících UCI

Další z hlavních výhod UCI je flexibilita a rozšiřitelnost. Rozhraní používá v rámci konfiguračních souborů přehledně definovanou datovou strukturu. Klíčové slovo config uvozuje v souboru novou sekci. Každá sekce má svůj vlastní jmenný identifikátor, který je buď specifikován uživatelem či konfiguračním příkazem nebo je vygenerován systémem. V rámci každé sekce je možné specifikovat pomocí klíčového slova option její parametry a hodnoty. Aplikace podporuje různé typy konfiguračních parametrů, včetně řetězců, celých čísel a seznamů, což pokrývá široký rozsah konfiguračních potřeb. Dále také umožňuje definici vlastních konfiguračních možností a validačních pravidel, což uživatelům dovoluje přizpůsobit konfigurace vlastním konkrétním požadavkům.[12]

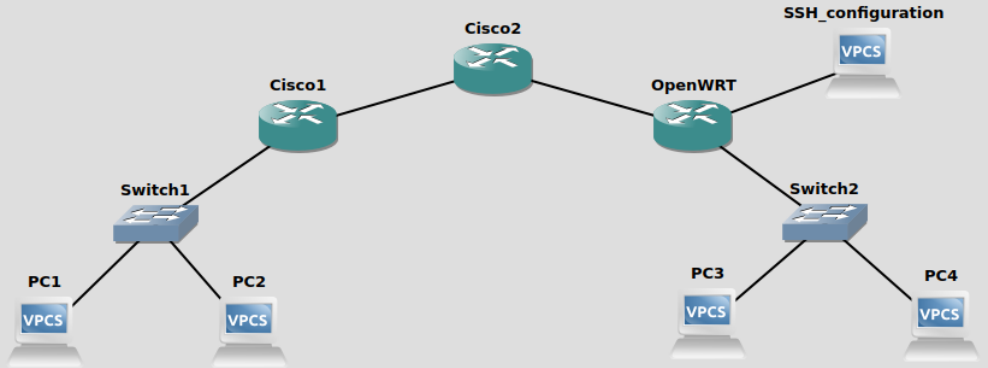
V rámci architektury OpenWRT slouží UCI jako základ pro konfiguraci nejen systémových nastavení, ale i zmíněných síťových rozhraní, firewall pravidel, a dalších vlastností sytému.

2.3 GNS3

Jako modelovací a testovací prostředí pro vyzkoušení emulační aplikace je používán virtualizační program GNS3. Prostředí GNS3 představuje open-source simulátor síťových zařízení, dostupný na operačních systémech Windows, Linux a macOS, který umožňuje vytvářet, konfigurovat a testovat základní i složité síťové situace ve virtuálním prostředí. Nástroj poskytuje prostředí pro emulaci a provozování různých síťových zařízení, včetně směrovačů, přepínačů, firewallů a dalších prvků, které jsou běžnou součástí reálných sítí. Uživatelé mohou vytvářet virtuální sítě za pomoci konfigurace daných síťových zařízení

pomocí běžných operačních systémů, které jsou dostupné v reálných zařízeních (např. Cisco IOS, Juniper JunOS) a simulovat jejich chování v různých scénářích. GNS3 podporuje integraci s reálnými sítěmi a umožňuje propojení vytvořených simulovaných sítí s reálnými pomocí technologií jako je Dynamips, VirtualBox či QEMU. Klíčovou vlastností pro výběr tohoto nástroje byla možnost importovat a spouštět vlastní vytvořené image operačního systému OpenWRT. [13]

Obrázek 2.4 demonstruje simulované síťové prostředí.



Obrázek 2.4 Schéma testovací vývojové sítě

2.4 Wireshark

Wireshark představuje open-source nástroj pro detailní analýzu veškerého síťového provozu v reálném čase. V rámci počítačových sítí a síťového monitorování je tento nástroj široce využíván, zejména pro množství nabízených funkcí, jako je filtrace nebo vyhledávání pouze určitého typu komunikace v rámci sítě a možnosti detailního prohlížení jednotlivých paketů až do úrovně samotných bytů. V kontextu této práce je nástroj často využíván pro analýzu chování Cisco směrovačů, k nastavení protokolů TCP/IP v rámci emulátoru.

2.5 Volba technologií

Při vybírání použitelných technologií i samotné implementaci aplikace byl brán zřetel zejména na omezené výpočetní a paměťové prostředky, kterými cílová zařízení, jako jsou směrovače, disponují. Z těchto důvodů panovala při vývoji snaha o maximální využívání prostředků a funkcionalit, které prostředí nabízí již v prvotní systémové konfiguraci, bez nutnosti doinstalovávat nebo vytvářet další balíčky.

2.5.1 Programovací jazyk a knihovny

K aplikaci jazyků typu C++, nebo dalších objektově orientovaných jazyků vyšší úrovně jako je Java, je ale nutné přidat další důležité knihovny a rozšířit dostupné funkcionality systému. Tyto nástroje ale často vyžadují mnoho paměťových prostředků, což může pro mnohá zařízení představovat nejen zmíněný paměťový ale také výkonnostní problém. Z těchto důvodů bylo při vývoji aplikace použití vyšších objektových programovacích jazyků vyloučeno.

Po analýze zmíněných dostupných možností byl jako hlavní programovací jazyk vybrán jazyk C, který umožňuje kontrolu systémových prostředků, lepší přístup k hardwaru a obsahuje veškeré další potřebné nástroje pro implementaci dané aplikace. Navíc oproti ostatním programovacím jazykům není potřeba instalovat žádné další rozšiřující balíčky, systém OpenWRT má již v sobě vestavěnou standardní knihovnu C musl. Další možnou alternativou je knihovna uclibc. Obě zmíněné knihovny jsou optimalizovány pro běh na zařízeních s omezenými prostředky. Po testech a analýze nabízených funkcí obou knihoven jsem pro vytvoření shellu zvolil musl.

Pro zajištění interakce s obsluhou v prostředí příkazového řádku používám v emulátoru C knihovnu Editline. Jako druhá alternativa byla brána v potaz i knihovna GNU readline, kterou běžně využívají standardní příkazoví interpreti jako bash či zsh. Knihovna disponuje další rozšířenou funkcionalitou oproti knihovně Editline. Pro potřeby aplikace poskytují obě knihovny všechny požadované datové struktury a funkce nutné pro vytvoření daných funkcionalit výsledného shellu, jako je například příkazová historie, funkce pro implementaci nápovědy nebo doplnění argumentů v rámci příkazové řádky. Tyto vlastnosti jsem ověřil při vytvoření minimalistické testovací verze shellu s použitím obou knihoven. Hlavním důvodem pro konečné zvolení a použití knihovny Editline jsou však menší nároky na systémové úložiště.

2.5.2 Využití technologie balíčků

Celý OpenWRT systém, až na samotné linuxové jádro, tvoří soubor jednotlivých instalačních balíčků. Koncept balíčku jsem v rámci tvorby emulátoru také zvolil pro výslednou kompilaci zdrojových souborů aplikace. Hlavní součástí každého balíčku je řídící soubor Makefile. Tento soubor představuje pro kompilátor klíčový prvek, protože obsahuje potřebné instrukce k celému procesu sestavování, seznam závislostí daného balíčku a informace o umístění zdrojových souborů, uložených v rámci adresářové struktury balíčku. Makefile také podporuje i dodatečné stažení zdrojových souborů z externích zdrojů gitlab repozitáře a zajistí vytvářené emulační aplikaci všechny potřebné vazby.[14]

Při vytváření výsledného obrazu celého systému OpenWRT včetně funkcionality Cisco emulace je nutné před zahájením závěrečné tvorby image v menu sestavovacího procesu přidat do procesu kompilace i můj emulační balíček. Systém poté na základě vzájemných závislostí jednotlivých balíčků určí

pořadí kompilace a zahrnutí balíčků a sestaví funkční obraz nově generovaného směrovače OpenWRT.

2.5.3 Další nástroje

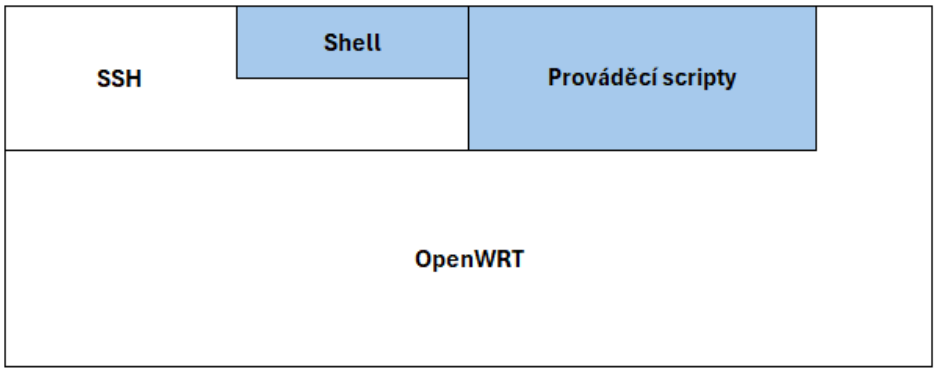
Pro tvorbu výkonných scriptů je výhodné využít vestavěný příkazový interpret ash, který nabízí všechny potřebné funkce.

Vzhledem k omezenému množství potřebných nástrojů je samotný vývoj celého emulátoru prováděn mimo cílové prostředí OpenWRT, na linuxové platformě v rámci distribuce Ubuntu, ve vývojovém editoru Visual Studio Code.

Kapitola 3
Implementace

Tato kapitola popisuje proces implementace emulovaného prostředí. Nejprve je popsán vývoj jednotlivých komponent emulátoru, v druhé části kapitoly je rozebírána implementace vybraných funkcionalit.

Emulátor se skládá ze dvou spolupracujících komponent. Hlavní viditelnou částí celého emulátoru je vlastní vytvořený shell, který interaguje s přihlášeným uživatelem a emuluje vizuální stránku příkazové řádky Cisco IOS. Jde o vytvořenou aplikaci spuštěnou po ověření uživatele a sestavení SSH kanálu. Zadané příkazy jsou v shellu podrobeny kontrole a příslušné modifikace chování směrovače provádí samostatně volané prováděcí scripty.



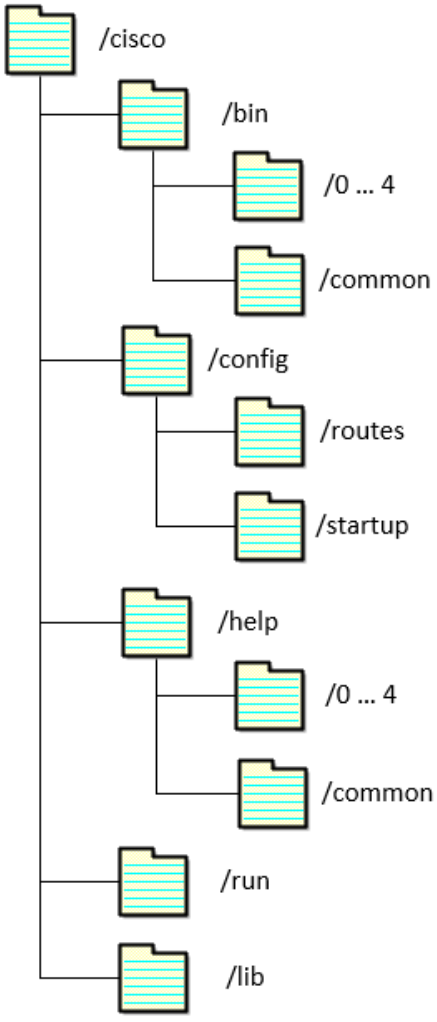
Obrázek 3.1 Architektura emulátoru

3.1 Datová struktura emulátoru

Pro uchování důležitých informací o stavu aplikace a hodnot proměnných využívá vytvořený systémový soubor. Tato data jsou načítána do prostředí shellu

vždy při vytvoření nového uživatelského spojení. V rámci aktivního využívání navázaného spojení se mohou hodnoty některých těchto proměnných měnit, proto shell po každém úspěšně provedeném příkazu kontroluje datum poslední modifikace souboru s proměnnými a při zjištění změny obnoví aktuální hodnoty uložených dat pro zachování konzistentního stavu aplikace. Shell také zodpovídá za správnost vložených příkazů uživatelem a kontroluje jejich syntaxi. V případě zjištění chyby generuje informační hlášku. Další funkcí je volání konfigurační scriptů, které modifikují chování směrovače OpenWRT.

Emulační aplikace spolupracuje s vytvořenou adresářovou strukturou, ve které jsou uloženy prováděcí a konfigurační scripty. Obrázek 3.2 demonstruje adresářovou strukturu emulátoru.



Obrázek 3.2 Adresářová struktura emulátoru

Kořenový adresář celého emulátoru představuje adresář /cisco. Shell v rámci vytvořené adresářové struktury přistupuje převážně do provozních adresářů bin a help. V každém z těchto adresářů jsou zrcadlově vytvořeny podadresáře, reprezentující vždy jeden implementovaný konfigurační mód (adresáře označené 0, 1, 2, 3, 4). Uvnitř těchto složek jsou poté uloženy všechny zásadní informace ohledně příkazů, které jsou v rámci originálního systému Cisco IOS v daném módu dostupné. Soubory jsou pojmenovány vždy podle jména každého emulovaného příkazu. Shell má pro správnou funkci vždy aktuální informace o konfiguračním režimu, ve kterém se uživatel v dané chvíli nachází a uživatele o tom informuje prompt zprávou v začátku příkazového řádku shellu. Ostatní složky slouží k uchovávání další důležité konfigurace a skriptů, které zajišťují korektní běh emulátoru. Složka config slouží pro ukládání nastavení emulátoru, jako je vlastní management statických směrovacích záznamů či startup konfigurace. Adresář run obsahuje spustitelné skripty a soubory, které jsou volány napříč různými příkazy, složka lib disponuje soubory s funkcemi, které je možné využívat v rámci skriptů.

3.2 Popis funkce shell emulátoru

Po obdržení příkazu od uživatele nejprve shell zkontroluje syntaxi a poté prohledává odpovídající konfigurační mód, reprezentovaný příslušným adresářem v adresáři help. Každý příkazový soubor v rámci tohoto adresáře obsahuje na samostatném řádku seznam všech možných parametrů, které mohou v rámci kontextu příkazu následovat, včetně jejich popisku, který odpovídá nápovědám v rámci operačního systému Cisco IOS. Některé parametry mohou obsahovat i speciálně vložené značky pro signalizaci shellu, že se jedná o parametr, který je nutné podrobit speciální sanitizaci, jako například IP adresa či číselný rozsah.

3.2.1 Validace

Pokud shell tedy nalezne soubor se jménem odpovídající názvu zadaného příkazu, načte soubor do paměti a postupným procházením tohoto souboru validuje jednotlivé vložené parametry do příkazové řádky. Narazí-li při čtení souboru s argumenty na speciální značku u některého z parametrů, zavolá shell příslušné vestavěné funkce pro sanitizaci zadaného parametru.

Vzhledem k tomu, že příkazový interpret Cisco IOS podporuje zadávání zkratek, je také vytvořený shell schopen vložené zkratky doplnit na kompletní znění daného parametru.

V případě zadání nevalidního vstupu je shell schopen, na základě vytvořené implementace, přesně určit pozici prvního chybného výskytu, stejně jako v Cisco IOS. Při kontrole jednotlivých parametrů zkouší shell namapovat daný argument, pomocí definovaných způsobů na jeden z možných validních parametrů na danou pozici. U každého neúspěšného pokusu si zapamatuje, na které pozici se od začátku argumentu objevila první neshoda a daný index si uloží.

V rámci tohoto procházení si shell pamatuje vždy nejdelší nalezenou shodu vůči validním argumentům. V případě, že žádný z validních parametrů neodpovídá vloženému vstupu na danou pozici v rámci příkazu, validace celého příkazu končí a shell zobrazí uživateli uložený index s příslušnou hláškou, kde při validaci příkazové řádky nastal problém.

Validace příkazu skončí chybou, i pokud uživatel zadá neexistující nebo pro systém nejednoznačný argument, avšak obsluha je i v těchto případech vždy korektně informován příslušnou chybovou hláškou o nalezeném problému v terminálovém okně.

V rámci této kontroly shell provádí pouze povrchovou prvotní validaci syntaxe jednotlivých argumentů daného příkazu, jako například u IP adresy či masky kontroluje pouze formát a reálnost vložených dat. Kontrola požadovaného obsahu parametrů je přenechána na volaných skriptech. Pokud jsou všechny argumenty zpracovávaného příkazu validní, soubor na každém konci možného příkazu obsahuje název funkce, která obsahuje prováděcí rutiny pro interpretaci daného příkazu.

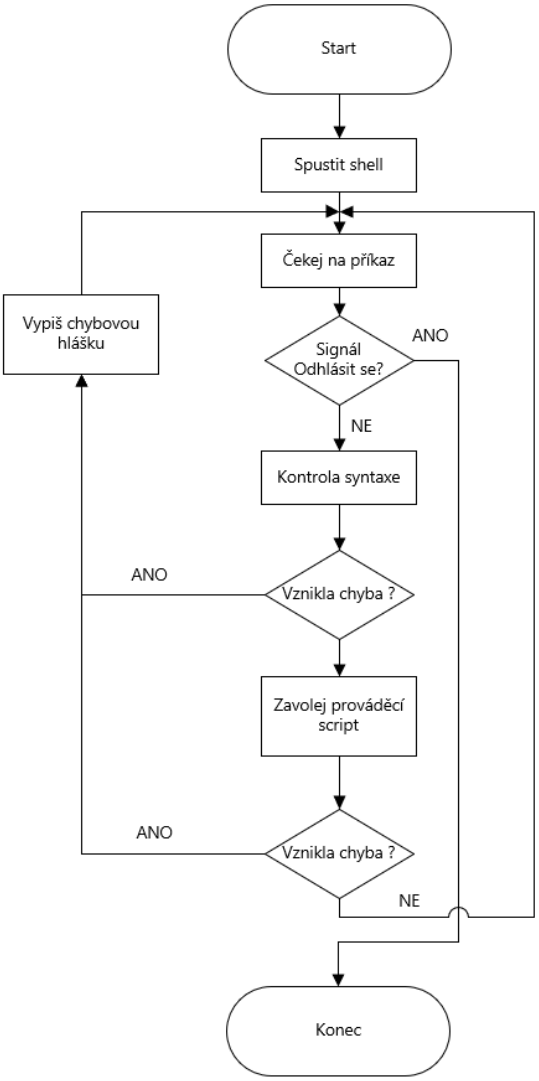
3.2.2 Prováděcí skripty

V příslušném podadresáři bin se v adresáři aktuálního konfiguračního módu nachází shell skripty, pojmenované též dle příkazů dostupných v rámci tohoto režimu. Na konci úspěšného vyhodnocování syntaxe příkazu vytvoří shell nový proces, ve kterém zavolá příslušný odpovídající skript a předá mu řízení aplikace. Celý proces chování shellu je graficky znázorněn na obrázku 3.3

Shell poskytuje těmto skriptům následující parametry: prvním parametrem je jméno funkce, která se má v kontextu příkazu zavolat, poté následuje kompletní příkazová řádka, kterou zadal uživatel. V případě, že byly pro zadání příkazu použity zkratky, zasílá vytvořený shell v rámci pozičních parametrů vždy celé znění všech argumentů.

Volané skripty provádí hlubší kontrolu přijatých parametrů a v případě chyby poskytují uživateli vlastní chybovou hlášku vypsáním do terminálového okna. Po obsloužení příkazu a případné modifikaci konfiguračních souborů v OpenWRT vrací skript řízení zpět shell procesu. Pokud skript modifikuje aktuální konfiguraci systému musí také sám volat příslušné funkce pro aplikování změněné konfigurace.

Systém skriptů a pomocných textových souborů byl zvolen zejména pro přehlednou práci s textovými soubory a možnosti snadného využití standardních unixových nástrojů. Pro spouštění těchto skriptů je využit vestavěný příkazový interpret ash. Skripty také zajišťují bezproblémové propojení s rozhraním UCI. Další výhodou je snazší další rozvoj a úprava systému, protože při změně konfiguračního skriptu není nutné znovu kompilovat celý balíček, jako v případě zdrojových souborů jazyka C.



Obrázek 3.3 Vývojový diagram vytvořeného shellu

3.3 Základní konfigurace obrazu OpenWRT

V rámci vytváření vlastní OpenWRT image nabízí kompilační systém možnost vkládat nové soubory, vytvářet vlastní adresářovou strukturu nebo upravovat obsah dosud neexistujících souborů systému OpenWRT ještě před prvním spuštěním systému. Pokud je pro účel vytvoření nové image použit nástroj ImageBuilder, cesta k adresáři, který obsahuje customizované soubory se do kompilačního procesu linkuje pomocí parametru FILES při zahájení kompilace. V případě kompilace přímo ze zdrojových kódů k tomuto účelu slouží vytvoření adresáře jménem files v hlavním adresáři celého kompilačního procesu.

3.3.1 Adresář files

Vytvořený adresář files je interpretován při kompilaci jako / (root) adresář v nově vznikajícím image. Soubory uložené v tomto adresáři nesmí být závislé na architektuře procesoru. Systém kopíruje všechny uložené tyto soubory přímo do vytvářené adresářové struktury vznikající image. V případě konfliktního pojmenování vložených souborů s generovanými soubory v rámci kompilace, upřednostňuje systém uživatelskou verzi. Možnost uložení vlastních souborů přímo do systému bez nutnosti jejich kompilace je v práci využita pro vytvoření vlastní adresářové struktury emulátoru a vkládání textových konfiguračních souborů emulátoru a shell skriptů obsahujících obslužné rutiny pro volané příkazy.[8]

3.3.2 Adresář uci-defaults

Další možností upravovat systémovou konfiguraci poskytuje adresář s názvem uci-defaults, nacházející se v adresáři /etc. Pro aplikaci požadovaných změn ihned po první spuštění systému je nutné adresář včetně jeho absolutní cesty vytvořit v rámci zmíněného adresáře files. Umístění vlastních skriptů do uci-defaults je vhodné zejména pro minoritní úpravy vzniklých systémových souborů bez nutnosti jejich celého přepsání a nastavení dalších potřebných vlastností systému před prvním použitím. Příkazy v rámci souborů tohoto adresáře jsou provedeny ihned po dokončení boot procesu systému v abecedním pořadí dle názvů jednotlivých souborů.[15]

Pro potřeby emulace jsou v adresáři /etc/config vytvořeny soubory pro konfiguraci NAT a ACL. Tyto soubory jsou upravovány ve skriptech pomocí nástroje UCI, jejich změna není navázána na žádnou systémovou službu, ale slouží pouze jako konfigurační soubory emulace.

3.3.3 Uživatelské účty

Systém OpenWRT neobsahuje v základní konfiguraci standardní linuxové nástroje pro manipulaci s autentizačními soubory, jako je /etc/passwd. Pro přidání nového uživatele je nutné tyto soubory s opatrností modifikovat ručně. V základní konfiguraci jsou všechna jména a hesla nastavena na slovo cisco.

3.4 První spuštění emulátoru

Při prvním spuštění směrovače jsou všechna rozhraní vypnuta, což emuluje výchozí stav Cisco směrovače. Výjimku tvoří pouze rozhraní Ethernet0, které po nabootování systému zůstává zapnuté a to pouze z důvodu umožnění navázání SSH spojení bez nutnosti předchozí konfigurace samotného směrovače jinými nástroji, než které jsou dostupné v rámci Cisco IOS. Tradiční zařízení Cisco je nutné poprvé konfigurovat pomocí hardwarového připojení na příslušný konzolový port daného zařízení. Tento krok nelze v rámci používaného prostředí nijak nasimulovat a je tedy nahrazen zmíněným zapnutým rozhraním.

Protokol SSH je v zařízení nakofigurován na standardní port 22 a s jeho využitím je možné bezpečně modifikovat a ovládat celé zařízení. Zapnuté rozhraní má nastavenou IP adresu 192.168.0.1/16. Pro spuštění emulačního prostředí Cisco směrovače je potřeba se do systému přihlásit pod identitou přidaného uživatele cisco a použitím stejnojmenného hesla.

Všechny provedené konfigurační změny se ukládají do běžící konfigurace, kterou v rámci emulace představují konfigurační soubory v systému. Pro zajištění shodného chování se systémem Cisco IOS jsou tyto soubory při restartu zařízení přepsány výchozí uloženou konfigurací.

Emulace v této verzi neposkytuje příkazy pro uložení aktivní konfigurace, ale obsahuje část implementovaných mechanismů a struktur, potřebných k zajištění této funkcionality v budoucím vývoji.

Funkčnost emulátoru po spuštění zařízení zabezpečuje také vytvořený init skript, který je v procesu bootování spouštěn jako poslední, poté co jsou nastaveny parametry a aktivovány systémové služby, důležité pro běh OpenWRT. Skript nejprve načte poslední uloženou konfiguraci směrovače do aktivních konfiguračních souborů systému a na základě této konfigurace upraví nastavení jednotlivých rozhraní. Skript je také schopen detekovat změny v počtu fyzických síťových rozhraní, dostupných na daném zařízení, a následně doplnit a upravit obsah konfiguračních souborů tak, aby bylo možné s novými rozhraními v rámci emulátoru nadále pracovat. V neposlední řadě zajistí aktivaci uložených pravidel pro filtrování paketů a nastaví příslušné proměnné prostředí klíčové pro emulátor.

3.5 Emulované funkce a příkazy

Ve vytvořeném emulátoru jsou zakomponovány všechny základní funkce nutné pro základní konfiguraci TCP/IP protokolu. Jejich výčet je součástí přílohy A.

Následující podkapitola detailně popisuje chování a implementaci pouze vybraných funkcionalit a jejich příkazů. Každá z funkcí je nejprve rozebrána a popsána v prostředí Cisco IOS a poté je okomentována její implementace v rámci emulace.

3.5.1 DHCP klient

DHCP je standardizovaný protokol sloužící k automatizaci procesu přidělování IP adres v síti a konfiguraci dalších důležitých parametrů nutných ke správnému fungování síťové komunikace, jako například určení defaultní brány. Typickou komunikaci DHCP služby mezi klientem a server tvoří následující zprávy: Discover, Offer, Request, Acknowledge.

3.5.1.1 DHCP v Cisco IOS

V operačním systému Cisco IOS je automatické přidělení IP adresy zajištěno pomocí příkazu *ip address dhcp* v konfiguračním módu vybraného rozhraní typu Ethernet. V rámci konfigurace DHCP klienta umožňuje IOS nastavení take dalších atributů, které klient posílá jako součást zpráv na server. Po vyjednání IP adresy provádí vzorová verze Cisco IOS kontrolu, zda obdržená adresa nekoliduje se sítěmi, které má zařízení nastavené na ostatních aktivních rozhraních.

Na obrázku 3.4 je zachycena testovaná komunikace mezi DHCP klientem a serverem v programu wireshark v případě, že právě serverem přidělená adresa leží v rozsahu některé přímo připojené sítě na jiném rozhraní, což má za následek obnovení celého přidělovacího DHCP procesu.

```
0.0.0.0             255.255.255.255     DHCP      618 DHCP Discover - Transaction ID 0x1a4
c2:08:63:8f:00:01   Broadcast           ARP        60 Who has 1.0.0.27? Tell 1.0.0.1
1.0.0.1             255.255.255.255     DHCP      342 DHCP Offer    - Transaction ID 0x1a4
0.0.0.0             255.255.255.255     DHCP      618 DHCP Request  - Transaction ID 0x1a4
1.0.0.1             255.255.255.255     DHCP      342 DHCP ACK      - Transaction ID 0x1a4
0.0.0.0             255.255.255.255     DHCP      618 DHCP Decline  - Transaction ID 0x1a4
0.0.0.0             255.255.255.255     DHCP      618 DHCP Discover - Transaction ID 0x1a4
c2:08:63:8f:00:01   Broadcast           ARP        60 Who has 1.0.0.28? Tell 1.0.0.1
1.0.0.1             255.255.255.255     DHCP      342 DHCP Offer    - Transaction ID 0x1a4
0.0.0.0             255.255.255.255     DHCP      618 DHCP Request  - Transaction ID 0x1a4
1.0.0.1             255.255.255.255     DHCP      342 DHCP ACK      - Transaction ID 0x1a4
0.0.0.0             255.255.255.255     DHCP      618 DHCP Decline  - Transaction ID 0x1a4
```

Obrázek 3.4 DHCP komunikace v případě konfliktu přidělených IP adres

Pokud DHCP server poskytuje v rámci konfiguračních parametrů také defaultní bránu, rozhodnutí o tom, zda je záznam použit ve směrovací tabulce, provádí systém na základě tzv. administrativní vzdálenosti. V případě že více směrovacích protokolů poskytuje cestu do stejné sítě, je vybrán protokol s nižší hodnotou tohoto ukazatele. Pro staticky definované směrovací záznamy má ukazatel hodnotu 1, pro defaultní bránu získanou z DHCP je výchozí hodnota 254. Obdržený záznam o výchozí bráně prostřednictvím DHCP je aktivní ve směrovací tabulce pouze tehdy, neexistuje-li cesta s nižší hodnotou administrativní vzdálenosti.

3.5.1.2 Emulace DHCP v OpenWRT

Systém OpenWRT používá pro potřeby DHCP klienta program udhcpc, který je součástí vestavěného nástroje Busybox.

Spuštění klienta zajišťuje systémová služba netifd, definující další parametry vytvářeného procesu, včetně hlavního skriptu volaného při jakékoliv zaznamenané DHCP události. Mezi tyto události patří nastavení nebo změna DHCP konfigurace na daném rozhraní a ukončení daného procesu. Pro každé aktivované DHCP rozhraní spouští systém vlastní proces. V případě potřeby poskytuje udhcpc uživatelům adresář určený pro uložení dalších konfiguračních skriptů, které tento proces spouští v abecedním pořadí vždy při zaznamenání některé ze zmíněných událostí. Všechny parametry jsou předávány ve formě proměnných prostředí. Pro potřeby emulace však nebylo využití pouze těchto možností dostatečné a bylo nutné modifikovat i samotný spouštěcí skript klienta. DHCP klient nekontroluje přidělené adresy vůči ostatním rozhraní a nastavuje adresu ihned po jejím obdržení, což může v nastavení daného zařízení způsobit nepředvídatelné chování, pokud je daná adresa konfliktní. Úpravy jsou zavedeny do skriptu před samotné nastavení adresy a kontrolují adresní rozsah všech aktivních rozhraní vůči získané IP adrese. V případě zjištění kolize je DHCP rozhraní ihned vypnuto. Pomocí použitých nástrojů nebylo možné dosáhnout úplně přesného chování jako na Cisco IOS. Udhcpc sice podporuje signály pro zasílání paketů na DHCP server, ale pouze pro obnovení či uvolnění dané IP adresy. Ani jeden z těchto signálů tak neumožňuje vyvolat stejné chování DHCP serveru jako v prostředí Cisco.

Další provedená úprava se týká vkládání záznamů o defaultních branách do směrovací tabulky, jelikož udhcpc proces automaticky aktivuje tyto záznamy ihned po nastavení IP adresy daného rozhraní. Aby bylo možné napodobit chování Cisco směrovače, vytvořený emulátor, na rozdíl od standardního chování udhcpc, dočasně ukládá záznamy o defaultních branách získané ze všech aktivních DHCP procesů na daném zařízení do určeného textového souboru. Po ukončení hlavního skriptu je volán vytvořený skript, který na základě nastalé DHCP události zkontroluje, zda uživatel již dříve manuálně nenastavil záznam o výchozí bráně spadající do sítě právě konfigurovaného rozhraní a případné nalezené záznamy přidá do směrovací tabulky. V případě, že zařízení nemá nastavenou žádnou aktivní výchozí bránu, aktivuje skript výchozí bránu z daného DHCP procesu.

3.5.2 Směrování

Korektní směrování paketů do jejich cílové destinace je základní prerekvizita pro funkční komunikaci mezi jednotlivými síťovými segmenty. Směrovací tabulky jednotlivých zařízení je možné využít dostupné dynamické směrovací protokoly nebo modifikovat tabulku přímo pomocí definované sady příkazů. V rámci vytvořeného emulovaného prostředí je umožněna pouze statická konfigurace jednotlivých směrovacích záznamů.

3.5.2.1 Směrování v Cisco IOS

Konfigurace statických záznamu ve směrovací tabulce Cisco zařízení je prováděna prostřednictvím příkazu *ip route <IP adresa> <maska> <brána>*. Prostředí umožňuje ukládat i údaje o cestách, kdy definovaná adresa brány není v momentě konfigurace dosažitelná přes žádné síťové rozhraní. Systém aktivuje a deaktivuje takto zapsané statické záznamy dle aktuální dosažitelnosti uvedené brány. Aktivní staticky definované záznamy mají při směrovacím procesu přednost před záznamy jiných směrovacích protokolů, pokud administrátor při jejich zadávání neuvede jinak. Pokud existuje pro cílovou destinaci více než jedna nejlepší zvolená cesta, provádí Cisco zařízení automaticky load balancing[1].

3.5.2.2 Emulace směrování v OpenWRT

Ukládání statických cest v systému OpenWRT je možné prostřednictvím příslušných konfiguračních souborů nebo přímým vložením do směrovací tabulky jádra. Při využití konfiguračních souborů vkládá systémová služba automaticky tyto záznamy do směrovací tabulky při znovunačtení obsahu souborů. Avšak pro vložení validního záznamu vyžaduje konfigurační soubor také určení konkrétního výstupního síťového rozhraní, které v momentě konfigurace záznamu nemusí být zřejmé. Přímé vložení záznamu do směrovací tabulky je možné pouze, pokud je v momentě konfigurace uvedená brána dosažitelná přes některé aktivní rozhraní, čímž jádro zajišťuje konzistenci tabulky. Pokud je ze směrovací tabulky smazán záznam o přímo připojené síti, odstraní se také všechny záznamy používající brány v této síti. Na rozdíl od Cisco IOS jsou ale takto odstraněné záznamy, namísto pouhé deaktivace, nenávratně ztraceny.

Pro emulaci statického směrování si aplikace uchovává informace o staticky definovaných cestách v rámci vlastní směrovací tabulky, která je vytvořena v rámci adresářové struktury emulátor v příslušném adresáři routes a dle uživatelem prováděných změn ve směrovací tabulce aktivuje či deaktivuje tyto záznamy. Při definování více záznamů o stejné síti jsou všechny přidány do směrovací tabulky, ale load balancing zajištěn není. Implementace této funkcionality vyžaduje větší systémové zásahy. Nabízí se možnost modifikovat chování směrovací tabulky v rámci samotného jádra před kompilací image. V rámci této konfigurace je tedy zajištěno pouze základní očekávané chování příkazů prostředí Cisco IOS bez možnosti load balancing.

3.5.3 NAT

Mechanismus překládání adres poskytuje možnost mapovat IP adresy z jednoho adresního rozsahu na druhý modifikací hlavičky průchozích IP paketů.

[1] rozložení zátěže

Existuje několik aplikovatelných technologií a způsobů, jak danou adresu přeložit. Vytvořená aplikace implementuje možnost dynamického překladu adres.

3.5.3.1 NAT v Cisco IOS

Konfigurace dynamického překladu adres na zařízeních Cisco vyžaduje použití následujících konfiguračních příkazů, které postupně definují jednotlivé parametry NAT. Ve výčtu jsou uvedeny verze příkazů, které je možné použít, jak v Cisco IOS, tak v emulovaném prostředí. Samotné Cisco IOS umožňuje u některých z těchto příkazů definovat ještě další parametry.

1. ip nat inside/outside
2. access-list permit <IP adresa> <wildcard maska>
3. ip nat pool <název poolu> <počáteční IP adresa> <koncová IP adresa> { prefix <1-32> | netmask <maska> }
4. ip nat inside source list <číslo ACL> pool <název poolu> [overload]

Cisco IOS vyžaduje přesné definování síťových rozhraní, které budou v procesu překládání figurovat. Rozhraní může být označeno z hlediska překladu jako vstupní nebo výstupní. Tato konfigurace probíhá v konfiguračním módu daného rozhraní a provádí se pomocí příkazu č. 1 s parametrem *inside* pro rozhraní připojená do LAN nebo *outside* pro rozhraní v rámci WAN. Pro určení rozsahu překládaných adres používá Cisco IOS systémové ACL v rámci příkazu č. 2. Příkazem č. 3 uvádí administrátor dostupné adresy, na které se adresy z LAN přeloží. Systém umožňuje takto nadefinovat několik ACL a NAT poolů[2], a proto v rámci korektní konfigurace NAT je potřeba systému specifikovat, který ACL a NAT pool má použít. To se provádí pomocí příkazu č. 4

V případě použití klíčového slova overload v příkazu č. 4, jsou adresy z definovaného privátního adresního rozsahu překládány na jednu vybranou veřejnou IP adresu. Rozlišení jednotlivých použitých překladů je zajištěno pomocí překladu zdrojových TCP/UDP portů uvnitř paketu v rámci PAT.

V opačné situaci, kdy parametr overload použit není, alokuje směrovač vždy jednu adresu, která je dostupná v rámci vymezené skupiny veřejných adres pro danou privátní adresu. V tomto případě je počet aktivních komunikací z vnitřní sítě směrem do vnější vždy omezen počtem v daný okamžik dostupných veřejných adres.

Popsané konfigurační kroky je možné provést v jakémkoliv pořadí, systém aktivuje NAT automaticky po obdržení všech potřebných informací.

[2] v rámci Cisco IOS definuje označení pro skupinu adres

3.5.3.2 Emulace NAT v OpenWRT

V systému OpenWRT je specifikace NAT i ACL uložena v rámci přidaných konfiguračních souborů, které obsluhuje systémový nástroj UCI. Nástroj UCI umožňuje efektivní procházení těchto souborů a přístup k jednotlivým položkám a je vhodný i pro použití ve skriptech systémových služeb. Při jakékoliv změně konfigurace těchto souborů, volá aplikace skript, který zkontroluje dostupnou konfiguraci příslušných souborů a na základě jejich stavu rozhodne, zda jsou splněny veškeré požadavky k aktivaci NAT či má být již aktivní NAT vypnut.

Samotná aktivace je v rámci emulace zajištěna pomocí vestavěného nástroje nftables. Jedná se o framework, dostupný v rámci linuxového jádra, který na základě vkládaných pravidel pomocí nástroje nft, umožňuje rozšířenou práci s IP pakety včetně jejich filtrace a modifikace jednotlivých atributů. Nftables ale při definici pravidel neposkytují možnost využití tzv. wildcard masky pro efektivní filtraci jednotlivých bitů v rámci IP adresy, jako tomu je v Cisco IOS. Emulační prostředí sice přijímá v rámci určených příkazů na vstupu wildcard masku, avšak do pravidel nftables ji interpretuje jako standardní síťovou masku. V případě, že při transformaci vznikne nesouvislá maska sítě, program bere tento stav jako chybný a NAT neaktivuje. Uživatelem poskytnuté dostupné adresy pro použití ve WAN v rámci aktivované NAT skupiny jsou kontrolovány na přítomnost nejen speciálních rozsahů jako například 127.0.0.0 či 224.0.0.0 - 255.0.0.0, ale také na přítomnost adres sítí a broadcast adres v jiných rozsazích. Zmíněné případy emulátor při aktivaci překladových pravidel ignoruje a nastavení neprovede.

Kapitola 4
Navazující práce

Vytvořená aplikace otevřela mnoho dalších témat, která lze dále rozpracovávat. Navazovat lze, jak v oblasti rozšiřování emulovaných funkcí, tak v oblastech zcela nových, v práci nezmíněných, a tak dále precizovat emulované Cisco prostředí. V této kapitole uvádím některá příklady.

4.1 Kontextová nápověda

Emulovaná aplikace v pracovním prostředí nepodporuje zobrazení kontextové nápovědy pro vkládaný konfigurační parametr vytvářeného příkazu. Použitá knihovna Editline, která zajišťuje ovládání a kontrolu příkazové řádky, umožňuje detekovat použití klávesové zkratky či specifického znaku v příkazové řádce. Této vlastnosti je možné využít a na vyvolanou událost navázat spuštění obslužné funkce v rámci shellu. Používaná datová struktura textových souborů, která definuje skladbu a syntaxi jednotlivých příkazů, již nyní u každého argumentu poskytuje jeho popis, který je shodný s nápovědou Cisco IOS. V rámci procesu procházení a validace jednotlivých argumentů, je možné tyto popisy extrahovat a nabídnout uživateli jako kontextovou on-line nápovědu.

4.2 Postup přidání dalšího příkazu

Množinu emulovaných Cisco příkazů lze dále jednoduše rozšiřovat což demonstruje následující popis. Jako ukázka byla vybrána implementace příkazu *ip nat inside source list <číslo ACL> pool <název poolu> [overload]*. Přidání tohoto příkazu je prováděno v několika krocích.

1. Zvolení příslušného módu, ve kterém má být daný příkaz dostupný. Na základě této volby zvolit příslušný adresář, reprezentující daný mód, do kterého bude příkaz vložen

2. Editovat soubory v rámci adresářů bin a help. Oba adresáře obsahují stejně pojmenované podadresáře, označující jednotlivé konfigurační módy
3. V adresáři help a vybraném podadresáři daného režimu je nutné upravit či vytvořit textový soubor nesoucí název příkazu (zde ip). Tento soubor definuje kontext a pořadí správného zadávání jednotlivých argumentů do příkazové řádky. Argumenty je tedy třeba rozdělit do jednotlivých úrovní pro zajištění správného kontextu a dále přidat příslušný popis každého argumentu. Pokud některý z argumentů požaduje speciální ošetření (formát IP adresy nebo kontrola číselného rozsahu), je nutné ho patřičně označit. Každý argument, kterým může zadávání příkazu končit, je třeba v souboru označit a uvést název funkce, která obsahuje funkcionalitu k obsloužení daného příkazu.
4. V rámci stejného zanoření, pouze v adresáři bin, se implementuje obslužná funkce pro daný příkaz. Shell těmto funkcím předává v rámci pozičních parametrů celé znění příkazové řádky.
5. Otestovat funkčnost přidaného příkazu

Z výše uvedeného postupu je zcela patrné, že vyvinutý emulátoru nepotřebuje pro rozšiřování sady jednoduchých příkazů protokolu IP verze 4 jakkoliv zasahovat do zdrojových kódů shellu, což výrazně zrychluje a zjednodušuje celý proces vývoje.

4.3 DHCP server

Další možností, jak rozšířit funkcionalitu emulovaného prostředí je DHCP server. Výchozí balíček v systému OpenWRT poskytující funkcionalitu DHCP serveru se nazývá dnsmasq. Nástroj implementuje statické i dynamické přidělování IP adres a dále i BOOTP protokol pro umožnění síťového bootování některých zařízení. Tato aplikace je schopna mimo DHCP službu poskytnout také základní funkcionalitu v oblasti DNS serveru.[16]

4.4 Dynamické směrování

Další možností je podpora dynamických směrovacích protokolů. U této funkcionality však z prvotní analýzy vyplývá, že korektní implementace této funkcionality bude vyžadovat větší systémové zásahy a kontrolu nad některými strukturami v rámci jádra. Využít lze například populární směrovací protokol OSPF, který je v rámci OpenWRT dostupný v balíčku frr-ospfd.

4.5 IP verze 6

Rodina příkazů protokolu IPv6 přestavuje v prostředí Cisco velkou oblast, kde lze rozšířit množinu emulovaných funkcionalit. Z provedené analýzy je zřejmé, že i na platformě OpenWRT je zmiňované oblast věnována velká pozornost. Při vývoji emulace se lze tak opřít o předpřipravená řešení ve formě balíčků a napodobit chování Cisco IOS i tomto poli.

Kapitola 5
Závěr

Bakalářská práce potvrdila, že lze vytvořit emulaci prostředí Cisco IOS v rámci open-source systému OpenWRT. Originální příkazy, jejich syntaxi i chování je možné plně nasimulovat. Vytvořený emulátor dovoluje řídit síťový provoz pomocí příkazů Cisco IOS i na jiných zařízeních než od primárního výrobce.

Emulovaná skupina příkazů pro řízení síťové komunikace zahrnuje konfiguraci síťových rozhraní, nastavení TCP/IP, směrování či DHCP klienta. Tyto funkcionality představují základní stavební kameny pro další rozšiřování množiny dostupných příkazů. Naprogramované prostředí využívá v maximální možné míře primárních vlastností OpenWRT včetně vhodně zvolené struktury práce s daty a nabízí platformu dovolující importovat prostředí i do vestavěných zařízení.

Vývojem aplikace jsem potvrdil, že kombinace obou technik ve formě kompilace C do binární podoby a využití skriptování ve vestavěném shellu představuje vhodné řešení, jak dosáhnout funkcionalit emulovaného prostředí, možné přenositelnosti na méně výkonná zařízení a zachování plné otevřenosti vzniklého prostředí. Při některých procesech emulace jsem však narážel na některá omezení, vycházející z platformy Linux, která bylo nutné složitě obcházet. Navržené řešení představuje základní funkční celek a nabízí prostor pro další úpravy a zlepšování celého systému.

Ve výsledku si lze představit, že je možné navázat na tento projekt a jeho dalším vývojem může vzniknout dokonale emulované prostředí Cisco IOS na platformě OpenWRT. Jde však o velmi časově náročnou činnost, která musí reflektovat mnoho vstupních požadavků a v průběhu dalšího vývoje bude nejspíš nutné modifikovat některé moduly jádra, které jsou spojeny s oblastí TCP/IP. Na závěr lze však říci, že emulace prostředí Cisco IOS v OpenWRT funguje.

Příloha A

Soupis a struktura emulovaných příkazů

1. access-list permit <IP adresa> <wild card maska>
   - vytvoření ACL povolující provoz definované sítě (použití pro NAT)
2. configure terminal
   - přepnutí uživatele z privilegovaného módu do globálního konfiguračního módu
3. disable
   - přepnutí uživatele z privilegovaného režimu do uživatelského režimu
4. do <příkaz>
   - umožní provádět příkazy dostupné v privilegovaném módu i v rámci konfiguračních módů
5. enable
   - přepnutí uživatele z uživatelského režimu do privilegovaného režimu
6. end
   - přepnutí uživatele z jakéhokoliv konfiguračního režimu do privilegovaného režimu
7. exit
   - opuštění aktuálního módu

8. hostname <jméno>
   - změní označení směrovače
9. interface { FastEthernet <0/0-N> | Loopback <0-M> }
   - přepnutí do konfiguračního módu zvoleného síťového rozhraní
10. ip address { <IP adresa> <maska> | dhcp }
    - nastaví IP adresu na zvoleném rozhraní (staticky nebo dynamicky pomocí DHCP)
11. ip name-server <IP adresa>
    - nastaví IP adresu dostupného DNS serveru pro DNS dotazy pro daný směrovač
12. ip nat { inside | outside }
    - označí zvolené rozhraní v rámci NAT konfigurace jako vnitří (LAN) nebo vnější (WAN)
13. ip nat inside source list <číslo ACL> pool <název poolu> [overload]
    - vytvoří NAT překladové pravidlo, které definuje adresy určené pro překlad, a specifikuje NAT pool, ze kterého se vyberou adresy, na které se bude v rámci NAT překládat
14. ip nat pool <název poolu> <start-ip> <end-ip> { prefix <1-32> | netmask <maska> }
    - vytvoří pool adres, který může být následně použit pro překlad adres
15. ip route <adresa sítě> <maska> <brána>
    - nastaví statický směr do dané sítě
16. logout
    - korektní ukončení aktivního spojeni
17. no <příkaz>
    - zruší konfiguraci příkazu v načtené konfiguraci
18. ping <IP adresa> [repeat <1-N> | timeout <1-M>]
    - zasílá ICMP pakety na danou IP adresu

19. show ip interface brief [<síť. rozhraní>]
    - zobrazí souhrn informací o všech/vybraném síťovém rozhraní
20. show ip interface <síť. rozhraní>
    - zobrazí detailní informace o vybraném síťovém rozhraní
21. show ip route
    - vypíše aktuální obsah směrovací tabulky
22. show running-config
    - vypíše veškerou dostupnou aktuální konfiguraci
23. shutdown
    - vypne příslušné síťové rozhraní
24. traceroute <IP adresa>
    - zasílá sérii paketů s postupně zvyšujícím se TTL směrem k zařízení s definovanou IP adresou v rámci příkazu

Bibliografie

1. ZOLA, Andrew; SCARPAT, Jessica. What is Cisco IOS (Cisco Internetwork Operating System)? [online]. 2023 [cit. 2024-05-02]. Dostupné z: https://www.techtarget.com/searchnetworking/definition/Cisco-IOS-Cisco-Internetwork-Operating-System.
2. ComputerNetworkingNotes: Cisco IOS Mode Explained with Examples [online]. 2024-01-08 [cit. 2024-05-04]. Dostupné z: https://www.computernetworkingnotes.com/ccna-study-guide/cisco-ios-mode-explained-with-examples.html.
3. DABLER. Příkazy Cisco IOS [online]. 2023-03-31 [cit. 2024-04-30]. Dostupné z: https://cs.wikibooks.org/wiki/P%C5%99%C3%ADkazy_Cisco_IOS.
4. OpenWRT Wiki: About the OpenWrt/LEDE project [online]. 2024-03-02 [cit. 2024-05-02]. Dostupné z: https://openwrt.org/about.
5. OpenWRT Wiki: Using the SDK [online]. 2022-04-21 [cit. 2024-05-07]. Dostupné z: https://openwrt.org/docs/guide-developer/toolchain/using_the_sdk.
6. OpenWRT Wiki: Build system setup [online]. 2024-04-28 [cit. 2024-05-05]. Dostupné z: https://openwrt.org/docs/guide-developer/toolchain/install-buildsystem.
7. OpenWRT Wiki: Using the ImageBuilder [online]. 2024-04-01 [cit. 2024-05-08]. Dostupné z: https://openwrt.org/docs/guide-user/additional-software/imagebuilder.
8. OpenWRT Wiki: Build system usage [online]. 2024-04-07 [cit. 2024-05-02]. Dostupné z: https://openwrt.org/docs/guide-developer/toolchain/use-buildsystem.
9. JIN, Tao. OpenWrt Development Guide [online]. 2012-02-13 [cit. 2024-04-27]. Dostupné z: https://www.khoury.northeastern.edu/home/noubir/Courses/CS6710/S12/material/OpenWrt_Dev_Tutorial.pdf.

10. OpenWRT Wiki: Build system essentials [online]. 2024-02-26 [cit. 2024-04-18]. Dostupné z: https://openwrt.org/docs/guide-developer/toolchain/buildsystem_essentials.
11. BusyBox: The Swiss Army Knife of Embedded Linux [online]. 2008 [cit. 2024-05-01]. Dostupné z: https://www.busybox.net/about.html.
12. OpenWRT Wiki: The UCI system [online]. 2024-02-23 [cit. 2024-04-30]. Dostupné z: https://openwrt.org/docs/guide-user/base-system/uci.
13. Galaxy Technologies LLC: Getting Started with GNS3 [online]. 2024 [cit. 2024-05-10]. Dostupné z: https://docs.gns3.com/docs/.
14. OpenWRT Wiki: OpenWrt packages [online]. 2021-10-15 [cit. 2024-04-18]. Dostupné z: https://openwrt.org/docs/guide-developer/package-policies.
15. OpenWRT Wiki: UCI defaults [online]. 2023-10-14 [cit. 2024-05-04]. Dostupné z: https://openwrt.org/docs/guide-developer/uci-defaults.
16. OpenWRT Wiki: Dnsmasq DHCP server [online]. 2022-05-12 [cit. 2024-04-21]. Dostupné z: https://openwrt.org/docs/guide-user/base-system/dhcp.dnsmasq.

Obsah přiloženého média

- `license.txt` ..... Licence přiložených zdrojových kódů
- `readme.txt` ..... stručný popis obsahu média
- `emulator.img` ..... image emulátoru
- `src`
  - `codes` ..... zdrojové kódy implementace v jazyce C
  - `cisco` ..... adresářová struktura emulovaného prostředí včetně skriptů
- `text` ..... text práce
  - `thesis.pdf` ..... text práce ve formátu PDF