České vysoké učení technické v Praze
Fakulta elektrotechnická
Katedra telekomunikační techniky

Bakalářská práce

Automatizace konfigurace síťových prvků pomocí Ansible

Miroslav Hudec

Studijní program: Komunikace, multimédia a elektronika
Studijní obor: Síťové a informační technologie
Vedoucí práce: Ing. Miloš Kozák, Ph.D.

Praha, Květen 2016

České vysoké učení technické v Praze
Fakulta elektrotechnická

katedra telekomunikační techniky

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

Student: **Miroslav Hudec**

Studijní program: Komunikace, multimédia a elektronika
Obor: Síťové a informační technologie

Název tématu: **Automatizace správy síťových prvků pomocí Ansible**

Pokyny pro vypracování:

Využijte nástroj Ansible pro automatickou správu sítových prvků. Zaměřte se na sítové prvky od společnosti Mikrotik a rozšiřte nástroj Ansible tak, aby bylo možné konfigurovat základní i pokročilé funkce těchto sítových prvků. Mezi základní funkce patří správa uživatelský účtů, logování a adresace sítových rozhraní. Mezi pokročilé funkce patří správa firewallu, překladu adres a sítových mostů. Navrhněte případy užití, na kterých ukážete funkci implementovaného rozšíření.

Seznam odborné literatury:

[1] Ansible: Up and Running, Lorin Hochstein, 2015. Dostupné na http://docs.ansible.com/ansible/ [on-line].
[2] *RouterOS by Example: Understanding MikroTik RouterOS Through Real Life Application*, Stephen R.W. Discher, 2011. Dostupné na http://wiki.mikrotik.com/wiki/Manual:TOC [on-line].

Vedoucí: Ing. Miloš Kozák, Ph.D.

Platnost zadání: do konce letního semestru 2016/2017

L.S.

prof. Ing. Boris Šimák, CSc.
vedoucí katedry

prof. Ing. Pavel Ripka, CSc.
děkan

V Praze dne 21. 12. 2015

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.

V Praze dne 27.5.2016

Miroslav Hudec

Poděkování

Abstrakt

V důsledku stálého rozvoje dnešních počítačových sítí a zvyšovaní nároků na ně kladených, je často potřeba zavádět do produkčního provozu větší množství nových a výkonnějších zařízení. V současnosti bývají tato zařízení konfigurována jedno po druhém, což prodlužuje dobu nutnou pro nastavení a tím samozřejmě zvyšuje náklady na implementaci. Cílem této práce je zefektivnění tohoto postupu navrhnutím řešení, které umožní automatizovanou konfiguraci mnoha zařízení dle zadaných požadavků. Zařízení by tak bylo možné nakonfigurovat během velmi krátké doby a s minimálním úsilím. V rámci této práce jsem se věnoval konfiguraci zařízeních MikroTik. Konfigurace síťových zařízení je realizována pomocí konfiguračního nástroje Ansible, který jsem doplnil o moduly umožňující nastavení základních funkcí se zaměřením především na prvotní konfiguraci související s nasazením většího počtu síťových prvků do sítě.

Klíčová slova

Automatizace, konfigurace, síťové prvky, MikroTik, RouterBOARD, RouterOS, Ansible

Abstract

Key words

Obsah

Seznam použitých zkratek

API Application Programming Interface 4, 17, 20–22

ARP Address Resolution Protocol 25

BGP Border Gateway Protocol 11

CDP Cisco Discovery Protocol 17, 28

CLI Command Line Interface *(Příkazová řádka)* 1, 4, 12, 13, 17, 22

DHCP Dynamic Host Configuration Protocol 4, 14, 18, 19, 21, 32

DNS Domain Name System 14, 32

DRP Disaster Recovery Plan 2

EIGRP Enhanced Interior Gateway Routing Protocol 11

IDPS Intrusion Detection and Prevention System 5

IP Internet Protocol 8, 11, 18

ISP Internet Service Provider *(Poskytovatel internetu)* 1, 10, 16

ITIL Information Technology Infrastructure Library 2

JSON JavaScript Object Notation 23, 28

LAN Local Area Network 18

MNDP MikroTik Neighbor Discovery Protokol 17

NAT Network Address Translation 18

OSPF Open Shortest Path First 11

RIP Router Information Protocol 11

SDN Software Defined Network 5

SNMP Simple Network Management Protocol 3, 4, 13

SOHO Small Office / Home Office 13

SPOF Single Point of Failure 7, 8

SSH Secure Shell 14, 20

TCP Transmission Control Protocol 8, 22

TFTP Trivial File Transfer Protocol 14

UDP User Datagram Protocol 8

VLAN Virtual Local Area Network 10

WAN Wide Area Network 19

YAML Yet Another Markup Language 15

1 Úvod

Rozvoj datových sítí v současné době probíhá velmi rychlým tempem. Poptávka po službách přenosu dat stále narůstá a datové sítě navíc přebírají role mnoha dalších sítí, jako jsou například hlasové či televizní sítě. Zvyšující se požadavky na propustnost a spolehlivost datových sítí spolu s rostoucím počtem internetových přípojek ústí v potřebu nasazovat velké počty síťových zařízení především do přístupové části sítě. Nasazování většího počtu síťových zařízení s sebou však přináší mnoho výzev. Nová zařízení je potřeba správně nastavit tak, aby odpovídala požadavkům dané sítě a umístění konkrétního prvku v topologii sítě. Dále je pak potřeba tato zařízení dohledovat, aby bylo možné odhalit případné poruchy či abnormality v síťovém provozu. V neposlední řadě je třeba umožnit vzdálenou správu zařízení tak, aby bylo možné provádět změny v konfiguraci každého zařízení v návaznosti na aktuální požadavky bez nutnosti fyzického přístupu k zařízení.

Ať už se jedná o nasazování aktivních prvků v síti Internet Service Provider *(Poskytovatel internetu)* (ISP) či v rozsáhlejší podnikové síti, vždy platí, že čas jsou peníze. A ekonomické hledisko bývá často až na prvním místě. Je proto velmi důležité celý proces implementace nových zařízení do provozu co nejvíce zefektivnit, a to jak po stránce potřebného času, tak nároků na lidské zdroje. V ideálním případě by tak měl celý proces implementace obstarat jeden technik v co nejkratším čase a samozřejmě bezchybně. K naplnění takových požadavků je však potřeba nalézt systém pro usnadnění a automatizaci celého procesu.

1.1 Správa moderních sítí

Korektně definovaná a aplikovaná správa počítačových sítí představuje fundamentální předpoklad pro jejich hladký a bezchybný provoz. Navzdory svojí důležitosti a nárokům kladeným na dnešní sítě však často administrátoři těchto sítí spoléhají na poměrně základní a přímočaré postupy: Změny v konfiguracích jednotlivých síťových prvků jsou převážně prováděny ručně pomocí zastaralých low-level nástrojů, jako je Command Line Interface *(Příkazová řádka)* (CLI). Takovýto přístup, který vyžaduje připojení se zvlášť ke každému zařízení je však velmi časově náročný. Navíc ve světě, kde jsou sítě stále rozlehlejší, komplexnější a dynamičtější roste i riziko lidské chyby. Takováto chyba v konfiguraci byť jediného zařízení přitom může mít fatální dopady na provoz celé sítě.

Prvotní pokusy o lepší zvládnutí obsluhy síťových prvků vzešly právě od síťových administrátorů, kteří sepisovali a spouštěli nejrůznější skripty. Tyto skripty se však postupem času stávaly stále sofistikovanější, přičemž každý administrátor udržoval svůj vlastní "arzenál". Takovéto skripty mají však bohužel velmi specifické zaměření a vyžadují vysokou

úroveň znalostí pro provedení složitějších úkonů a zamezení nechtěným vedlejším účinkům. Sestavení takového skriptu navíc vyžaduje značnou odbornost a dlouhé testování metodou pokus-omyl.

Dalším nezbytným předpokladem pro provoz počítačové sítě je správa dokumentace, především udržování záznamů o stavu jednotlivých prvků a jejich konfigurací. Dojde-li například k poruše zařízení, je třeba zajistit, aby po opravě poruchy bylo možné uvést zařízení do původního stavu v co nejkratším čase. Za tímto účelem bylo vyvinuto mnoho softwarových řešení, které umožňují automatické zálohování konfigurací a jejich následnou obnovu v případě poruchy. Pro ochranu podnikové IT infrastruktury by také měl existovat Disaster Recovery Plan (DRP), který přesně definuje procesní postupy pro případ havárie.

1.1.1 Centralizovaná správa

Aby bylo možné naplnit veškeré požadavky na provoz sítě, je často nutné celou její správu centralizovat. Jsou-li veškeré údaje o stavu sítě dostupné z jednoho kontrolního místa, je mnohem snazší dohlížet na provoz, odhalit případné poruchy a provádět změny v konfiguraci sítě s ohledem na všechny důsledky. Je velmi důležité, aby každá změna byla provedena s ohledem na všechny návaznosti a následně řádně zdokumentována. Provádění změn v síti tak říkajíc ad hoc, například na základě požadavků uživatelů bez jejich řádných zavedení do dokumentace sítě může vést k nestandardnímu chování celé sítě. Toto pak zvláště platí v případě, že o síť se stará více administrátorů. Aby se zamezilo takovýmto možným nedopatřením, měla by být v rámci správy sítě zavedena a striktně dodržována metodika procesního řízení. Jednou z takovýchto standardizovaných metodik je například Information Technology Infrastructure Library (ITIL). Jedná se o sadu praktik pro správu IT služeb, které se zaměřují na sladění IT s potřebami podniku [1]. Takováto metodika by poté měla být součástí širší podnikové politiky.

Monitoring počítačové sítě: Monitoringem počítačové sítě se rozumí nepřetržité sledování stavu zařízení v síti, které umožní odhalit selhávající komponenty sítě a upozornit na tuto skutečnost administrátora sítě. Lze tak jednak předejít možným výpadkům v provozu či určit kde k poruše došlo a značně tak zkrátit čas potřebný na její odstranění. Jedním z jednodušších přístupů k této problematice je centrální shromažďování logů z jednotlivých zařízení. Každé zařízení většinou ukládá informace o svém stavu a provedených úkonech do souboru označovaného jako log. Tyto logy lze pak shromažďovat na jednom centrálním místě, kterým bývá log server. Logy na tomto serveru jsou poté automaticky vyhodnoceny a na základě jejich závažnosti je provedena předdefinovaná akce, jako například upozornění administrátora prostřednictvím e-mailu či SMS.

Další možností pro získávání informací o stavu zařízení je implementace Simple Network Management Protocol (SNMP). Jedná se o protokol navržený pro shromažďování a organizaci informací ze spravovaných zařízení. Většina dnešních zařízení, jako jsou routery, switche, servery, koncové stanice či tiskárny podporuje tento protokol, SNMP je proto hojně využíván v systémech síťové správy a jedná se de facto o standard. Informace ze zařízeních jsou zde prezentována pomocí proměnných, které mohou být vyhodnocovány prostřednictvím softwaru, který je spuštěn na SNMP serveru. Hodnoty jako například aktuální vytížení procesoru či šířka využitého pásma lze následně vykreslit do grafů, což poskytuje administrátorovi dobrý přehled o stavu jednotlivých zařízení. Stejně jako v předchozím případě je i zde žádoucí předdefinovat akce pro klíčové hodnoty, při jejichž dosažení dojde k informování administrátora. Použití SNMP tak administrátorům poskytuje mnohem sofistikovanější a flexibilnější řešení než prosté shromažďování logů. Další výhodou tohoto řešení je také to, že některá zařízení umožňují i změnu svojí konfigurace prostřednictvím SNMP.

Jednou z oblastí monitorování sítě je také sledování samotného síťového provozu. Informace o provozu směrem do a ze sítě administrátorům většinou zprostředkuje router či firewall a o stavu zařízení v síti jsou informováni díky SNMP. Někdy je ovšem vhodné monitorovat i provoz uvnitř lokální sítě, což poskytuje administrátorům mnoho dalších informací za účelem optimalizace sítě či odhalení případných chyb. Monitorování vnitřního provozu navíc zvyšuje bezpečnost, protože díky němu lze odhalit nestandardní síťový provoz, který by mohl představovat potenciální rizika. Za tímto účelem se využívají takzvané síťové sondy, které shromažďují a analyzují veškerý síťový provoz. Získané statistiky pak poskytují informace například o tom, jaký druh provozu ve kterých částech sítě a v jakou dobu prochází. Takových údajů lze využít pro potřebné změny v konfiguraci jednotlivých aktivních prvků pro optimalizaci provozu. Síťové sondy jsou také schopny na základě behaviorální analýzy odhalit odchylky od standardního provozu a upozornit na ně administrátora sítě.

Centrální správa konfigurace: K provozování počítačové sítě neodmyslitelně patří správa konfigurací síťových prvků. Nastavení síťových zařízení je ve většině případů reprezentováno souborem obsahujícím konfigurační příkazy. Existuje mnoho možností, jak tyto konfigurace zálohovat a v případě potřeby obnovit do dřívějšího stavu. Nejzákladnějším, ale nezřídka využívaným způsobem je prosté kopírování textu z příkazové řádky a následné uložení do textového souboru. Poněkud lepším řešením je přímo kopírování konfiguračního souboru například na FTP server. Pro rozlehlejší sítě s velkým počtem aktivních prvků je však velmi nevýhodné, aby tyto zálohy byly prováděny ručně. Za tímto účelem existuje řada komerčních nástrojů, jejichž účelem je usnadnit a urychlit proces zálohy a obnovy. Jedná se o aplikace, které se periodicky připojují k síťovým zařízením a kontrolují stav jejich konfigurace, provádějí její zálohu a v případě potřeby jsou schopny na daném zařízení obnovit nastavení do dřívějšího stavu.

Provádění záloh konfigurací síťových prvků je důležité z mnoha důvodů. Tím prvním je samozřejmě snaha minimalizovat dobu výpadku sítě, dojde-li k poruše zařízení. V takovém případě je nejrychlejším řešením připojit na místo původního zařízení zařízení nové a nahrát do něj konfiguraci ze zálohy. Stejně tak v případě, že síť začne vykazovat nestandardní chování, může být velmi rychlým a efektivním řešením problému vrácení se do stavu, kdy vše fungovalo korektně. Téměř vždy totiž platí, že pokud něco přestane fungovat, předcházela tomu nějaká změna. To je také dalším důvodem, proč pravidelně zálohovat konfigurace aktivních prvků. Je-li naplánovaná jakákoliv změna v konfiguraci sítě, provedení této změny by vždy měla předcházet záloha současného stavu. I při sebevětší snaze síťových administrátorů počítat se všemi možnými následky plánované změny se vždy může objevit něco, s čím se nepočítalo. V takovém případě je nezbytné v první řadě obnovit provozuschopnost sítě, a až následně se věnovat zjišťování co způsobilo daný problém a jak jej pro příště odstranit.

1.1.2 Automatizace počítačových sítí

Hlavním důvodem pro automatizování sítí je odlehčení zátěže administrátorům, kteří jsou jinak často nuceni trávit mnoho času při plnění rutinních a opakujících se úkonů. Administrátoři pak mohou věnovat více pozornosti plánování rozvoje stávající infrastruktury a jejího zefektivnění. Ve výsledku je pak díky automatizaci možné snížit počet administrátorů, což vede ke značnému snížení prostředků nezbytných pro provoz IT infrastruktury. Pojmem automatizace počítačové sítě se rozumí použití aplikací a nástrojů, které jsou schopny do jisté míry samostatně provádět komplexní, zdlouhavé či opakující se procesy při správě počítačové sítě ať s žádnou, či minimální interakcí ze strany administrátora sítě. Automatizované úkoly mohou provádět jak jednoduché a jednoúčelové skripty, tak robustní automatizační nástroje. Síťová automatizace však není pouze skriptování [2]. Zde jsou některé typy síťové automatizace:

Skriptově-orientovaná automatizace: Administrátoři píší skripty za účelem automatizace změny konfigurace síťových zařízení. Za tímto účelem se využívá například RESTful Application Programming Interface (API), YANG, NETCONF, či jednoduchých skriptů ovládajících CLI nebo SNMP. Tyto skripty jsou nejčastěji v jazycích Python, Puppet či Chef, ale lze samozřejmě použít i jiné jazyky. Inteligence je pak obsažena v těchto skriptech. Mnoho síťových prvků také podporuje API, které umožňují lépe algoritmizovat a uchopit přístup k zařízením.

Automatická konfigurace a provisioning: Některé automatizační schopnosti jsou přímo vestavěny do zařízení. Mnoho z nich je dnes považováno za standard, avšak začínaly jako jako automatizační nástroje. Typickým příkladem může být Dynamic Host Configuration Protocol (DHCP) server, díky kterému není třeba nastavovat statické adresy na klientských

zařízeních. Jedná se o tak základní službu, že na ni ani není pohlíženo jako na automatizaci, avšak stala se tak fundamentální, že si dnes bez ní deployment koncových zařízení lze jen těžko představit.

Automatický provoz a řízení: Automatizace pomáhá zejména s každodenními úkoly, jako je reagování na události a následné rekonfiguraci zařízení. Jakýkoliv úkol z kategorie "prozkoumat a reagovat"spadá sem. Do této kategorie mohou spadat například síťové bezpečnostní systémy jako Intrusion Detection and Prevention System (IDPS), které fungují jako automatizované senzory síťového provozu schopné provést adekvátní akci na základě povahy provozu, například ukončit spojení.

Orchestrace a virtualizace: Další úrovní automatizovaných sítí je koncept tzv. softwarově definovaných sítí - Software Defined Network (SDN). Jedná se o koncept virtualizace kompletní síťové infrastruktury a její koordinované spolupráce. Existuje mnoho definic SDN, avšak v základu se jedná o konceptuální oddělení Data plane a Control plane. Toto rozdělení poskytuje široké možnosti konfigurace jednotlivých prvků. To závislosti na schopnostech takového systému může vést k aplikacemi řízeným sítím, což znamená, že lze nahrávat aplikace poskytující síťové funkce do ovladače (controlleru) a poskytovat tyto funkce v rámci celé sítě. Před příchodem SDN museli členové IT oddělení nasadit nový firmware či dokonce nový hardware, aby mohli poskytovat funkce. S příchodem SDN může být chování sítě řízeno aplikacemi. Společnost Hewlett-Packard kupříkladu provozuje SDN App Store, kde poskytuje aplikace řídící síťová zařízení podporující OpenFlow. Existují také opensourcové SDN systémy, které jsou modulární a jejich schopnosti lze rozšiřovat formou aplikací. Softwarově definovaná síť je pak schopna automaticky přizpůsobovat své chování aktuálním požadavkům v reálném čase a poskytovat tak vyšší kvalitu služeb koncovým uživatelům.

Politikou řízená síť: Tato forma softwarově definované sítě funguje na principu deklarování požadované politiky. Správce sítě popíše *co* chce v sítí provést, a systém sám určí *jak* tyto změny provést. Jedná se o pokročilou formu síťové automatizace, která umožňuje například provozovatelům aplikací definovat, jaké chování od sítě očekávají. Jako příklad lze uvést *Application Centric Networking* od společnosti Cisco [3].

1.2 Zaměření a cíl práce

Cílem této práce je navrhnout a realizovat postup pro automatizovanou konfiguraci základních funkcí na síťových zařízeních MikroTik s využitím nástroje Ansible. Primárně je tato práce zaměřena na takzvanou out-of-box konfiguraci, tedy prvotní nastavení zařízení, které umožní jeho nasazení do sítě. V rámci této práce se budu věnovat možným postupům k dosažení tohoto cíle, následně popisu mnou zvoleného řešení a jeho obhajobou. V závěru práce pak

budou uvedeny konkrétní případy použití. Cílem této práce však není vytvoření plně automatizovaného nástroje pro kompletní správu síťové infrastruktury vystavěné na zařízeních společnosti MikroTik, nýbrž především co nejvíce usnadnit nastavení základních síťových funkcí a umožnit tak následnou správu pomocí dalších nástrojů. Vzhledem k modulární povaze nástroje Ansible bude však možné v budoucnu přidat další funkce a rozšířit tak schopnosti tohoto nástroje.

2 Analýza

2.1 Základní pojmy

2.1.1 Charakteristiky počítačové sítě

Mezi základní charakteristiky počítačové sítě patří: Topologie, rychlost, cena, bezpečnost, dostupnost, škálovatelnost a spolehlivost.

Topologie: Pod pojmem topologie si lze představit určitý tvar či strukturu sítě. Zabývá se propojením síťových prvků v rámci sítě a zachycením reálné (fyzické) a logické podoby tohoto propojení. Fyzická topologie popisuje skutečnou konstrukci sítě s jednotlivými uzly a fyzickými zařízeními, včetně popisu kabeláže, která je propojuje. Logická topologie se pak vztahuje k tomu, jak jsou v síti přenášena data mezi jednotlivými uzly. Ačkoliv logická topologie může kopírovat topologii fyzickou (například v domácích sítích), ve větších sítích se tyto topologie liší. Důvodem je zejména zvýšení dostupnosti, spolehlivosti a tím celkové robustnosti sítě. Dojde-li například k poruše zařízení či závadě na kabelu, je možné pozměnit logickou topologii při zachování té fyzické. Provoz bude veden jinou cestou a síť bude stále dostupná a provozuschopná. Změnu logické topologie v případě poruchy jsou pak zařízení schopna provést sama bez vnějšího zásahu, s využitím zvláštních protokolů a ve velmi krátkém čase. S možností poruchy je třeba počítat již při návrhu fyzické topologie a pokud možno zamezit vzniku Single Point of Failure (SPOF), tedy bodu, jehož selhání povede k výpadku provozuschopnosti sítě.

Rychlost: Rychlost sítě udává, jaké množství informace je síť schopna přenést za určitý čas, základní jednotkou rychlosti je bit za sekundu (b/s, bps). Dnešní lokální sítě většinou využívají v přístupové části rychlostí 100 Mb/s označovanou jako Fast Ethernet, nebo 1000 Mb/s označovanou jako Gigabit Ethernet. Vzhledem k neustále rostoucímu objemu přenášených dat, jako sledování on-line videí, videokonferencí, přenos souborů či zálohování na síťová úložiště, rostou i požadavky na přenosovou rychlost v počítačových sítích. Speciálním případem jsou pak datové centra, která vzhledem k enormnímu objemu přenášených dat kladou vysoké nároky, mimo jiné, na přenosovou rychlost. To samozřejmě vyžaduje využití odlišné infrastruktury než u uživatelských sítí, například využití optických vláken, která dosahují vyšších rychlostí než metalické kabely.

Dostupnost: Dostupnost je parametrem popisujícím jak často jsou služby a prostředky poskytované počítačovou sítí přístupné uživatelům. Vetšinou se dostupnost uvádí jako množství času za rok, kdy byly služby poskytované sítí dostupné. Tento poměr je udáván v procentech,

kupříkladu dostupnost 99,99% představuje povolený výpadek sítě na 52,56 minut za rok, tedy asi 8,6 sekundy denně. Kritické systémy používané například bezpečnostními složkami, zdravotnickými zařízeními či datacentry vyžadují vysokou dostupnost *(High-availability)*. U takovýchto systému se často se uvádí dostupnost 99,999% (”pět devítek), což dává prostor pro nedostupnost služby po dobu 864,3 milisekund denně. K dosažení takto vysoké dostupnosti je třeba v síti či systému eliminovat SPOF, což vyžaduje redundanci a monitorování poruch v reálném čase. Vysoká dostupnost je velmi důležitá také pro podniky. Ve zprávě z roku 1998 společnost IBM uvádí, že americké podniky přišly v důsledku nedostupnosti svých systémů za rok 1996 o 4,54 miliardy dolarů[4].

Spolehlivost: Spolehlivost popisuje schopnost počítačové sítě provést požadovanou akci, jako například komunikaci mezi dvěmi zařízeními. Pro učení spolehlivosti sítě lze použít deterministické či pravděpodobnostní modely [5]. Spolehlivost v počítačové síti určuje jednak kvalita infrastruktury a použité komunikační protokoly. Některé síťové protokoly jsou označovány jako spolehlivé, neboť zahrnují funkce pro potvrzení o doručení vyslané zprávy, či jsou schopné zajistit její opětovné vyslání v případě chyby. Typickým spolehlivým protokolem je například Transmission Control Protocol (TCP). Naopak nespolehlivé protokoly nezaručují doručení vyslané zprávy, mohou se však hodit pro některé specifické aplikace. Asi nejznámější protokolem z této kategorie je User Datagram Protocol (UDP). Protokoly TCP i UDP jsou protokoly 4. vrstvy a využívají Internet Protocol (IP), který spadá do kategorie nespolehlivých protokolů.

Bezpečnost: Dnešní sítě a obecně počítačové systémy přenášejí a uchovávají mnoho citlivých informací. Nezbytným požadavkem při provozování sítě je proto zajištění její bezpečnosti, tedy snaha co nejvíce omezit možnost přístupu nepovolaným osobám k těmto datům. Do síťové bezpečnosti však nespadá pouze obrana proti nejrůznějším útokům, ale také omezení dopadu dalších potenciálních hrozeb. Mezi tyto hrozby patří například nedbalost či neznalost uživatelů, přírodní faktory jako požár či záplavy a další. Před plánováním zabezpečení sítě je pak třeba zvážit cenu aktiv (hardware, software, data) a jaké dopady může mít jejich odcizení, poškození či kompromitování. Bezpečnostní opatření by měla být dostatečná, nikoliv však přemrštěná (pro příměr - aby nebyla cena trezoru vyšší než cena jeho obsahu). V návaznosti na možný výskyt bezpečnostních incidentů je třeba mít připraven plán na jejich zvládání a minimalizování jejich dopadu.

Síťovou bezpečnost lze rozdělit do tří základních pilířů: fyzická bezpečnost, bezpečnost sítě a služeb a bezpečnost lidských zdrojů. Do fyzické bezpečnosti spadá například zajištění vhodného prostředí pro provoz IT infrastruktury a jeho zabezpečení proti přístupu nepovolaných osob (uzamykatelné racky a kabinety). Do této kategorie dále spadá fyzická topologie a redundance klíčových cest, provádění záloh důležitých dat či jejich ukládání na

bezpečných zrcadlených datových úložištích a další. Bezpečnost sítě a služeb se zabývá především zabezpečením přístupu k těmto službám prostřednictvím autentizace a autorizace, tedy ověřením uživatelů a nastavením přístupových práv k jednotlivým službám a logováním jejich činnosti. Spadá sem také prioritizace služeb, rozdělení sítě (bezpečnostní zóny, virtuální sítě), obvodová bezpečnost (Firewall, obrana proti útokům z internetu), ochrana koncových zařízení (Antivirové programy, IPS, IDS), ochrana sítě před uživatelem (povolení přístupu do sítě pouze ověřeným uživatelům a stanicím), používání zabezpečených služeb a šifrování, celková správa sítě (správa konfigurací zařízení, monitoring) a mnoho dalších. Bezpečnost lidských zdrojů zahrnuje mimo jiné informovanost uživatelů formou školení, minimalizaci přístupových práv (každý uživatel má přístup pouze tam, kam potřebuje), dodržování interních předpisů a politik. Bohužel právě tento aspekt síťové bezpečnosti bývá často opomíjen. Mnohdy platí, že nejslabším článkem v zabezpečení sítě je právě člověk. Na tuto skutečnost spoléhají útoky založené na sociálním inženýrství, kdy útočník získá klíčové informace od zaměstnance často pouhým dotazem. Útočník s připravenou záminkou kontaktuje některého zaměstnance a požádá jej o určité informace či provedení nějaké akce. Zaměstnanec pak v dobré víře udělá, co útočník žádá, aniž by si byl vědom možných následků svého jednání.

Cena: Jedním z hlavních parametrů při návrhu a provozu síťové infrastruktury je samozřejmě cena navrženého řešení. Hlavní podíl na koncové ceně však nemá pouze použitý hardware, ale především licence pro jednotlivá zařízení a jejich následná podpora ze strany výrobce. S provozem IT infrastruktury jsou také neodmyslitelně spojené náklady na provoz, tedy především platy zaměstnanců IT oddělení. To samozřejmě platí především u středních a velkých společností, které provozují rozsáhlé infrastruktury. Často je však na IT oddělení pohlíženo na jakýsi nadstandardní luxus. Pokud vše funguje bez problémů, připadá členům vedení neekonomické provozovat IT oddělení. Což je poněkud ironické, protože vše funguje bez problémů právě tehdy, když pracovníci IT oddělení odvádějí dobře svoji práci. V posledních letech se tak rozmáhá trend outsourcingu správy IT infrastruktury. Podniky pak mají možnost zredukovat IT oddělení až na jednoho člověka, který se většinu času zabývá například požadavky a problémy uživatelů. Tohoto trendu si samozřejmě všimli i velcí výrobci na poli IT a upravili tomu i své obchodní politiky. Dnes snad všichni velcí hráči poskytují také služby pro správu a dohled počítačových sítí. Jako příklad můžeme uvést společnost Hewlett-Packard, jejíž příjmy v roce 2015 tvořili ze 20% právě služby spojené s podporou a outsourcingem. Těchto 20% přitom představuje zhruba 20 miliard USD. Podnikům tak outsourcing správy jejich IT infrastruktury umožňuje snížit náklady spojené s provozem vlastních IT oddělení a platit externím společnostem pouze za to, co v danou chvíli potřebují. Pracovníci externích společností navíc často dosahují vyšší úrovně znalostí.

Škálovatelnost: Při návrhu síťové infrastruktury by se vždy mělo počítat s jejím růstem. Je nakonec cílem asi každého podniku zvyšovat své zisky, což je často spojeno se zvýšením

počtu zaměstnanců, což zase vede k potřebě více serverů, více aktivních prvků atd. Přístup "Aktuálně máme sedm zaměstnanců a jeden server, tudíž nám naprosto stačí osmi-portový switch"tak sice poskytuje zdánlivou úsporu při prvotním budování sítě, avšak z dlouhodobého hlediska může být značně nehospodárný. Přibude-li ve společnosti z výše uvedeného příkladu pozice pro dalšího zaměstnance, bude potřeba vyměnit původní switch za nový, poskytující více přístupových portů. Síť by tak vždy měla být navržena s určitou, nikoliv však přemrštěnou, rezervou. Toto samozřejmě neplatí pouze pro podnikové sítě, ale kupříkladu také pro sítě ISP, jejich primárním cílem je rozšiřovat svou síť a poskytovat své služby dalším zákazníkům.

2.1.2 Základní síťové prvky

Přepínač (Switch) Přepínač je síťově zařízení pracující na L2, tedy spojové vrstvě Referenční model ISO/OSI (RM ISO/OSI). Jedná se většinou o zařízení s více porty, kdy je ke každému portu připojeno koncové zařízení. Úkolem switche je pak předávat příchozí ethernetové rámce na příslušný port podle cílové MAC adresy. Switch si vede tabulku, ve které přiřazuje adresy zařízení připojených ke konkrétním portům.

Přepínače nahradili v sítích dnes již zastaralé rozbočovače neboli Huby. Ty se od switchů lišily především v tom, že si nevedly tabulku adres připojených zařízení, ale pouze rozesílaly příchozí rámce na všechny porty s výjimkou příchozího. To velmi zvyšovalo provoz v síti. Představme si, že máme 10 osobních počítačů připojených prostřednictvím rozbočovače. Bude-li chtít počítač č. 1 komunikovat s počítačem č. 2, veškerá jejich komunikace v obou směrech bude zároveň odesílána na počítače č. 3-č. 10. Ty pak tato data v lepším případě zahodí, protože jim nejsou určena, v horším případě to umožňuje potenciálnímu útočníkovi odposlouchávat tuto komunikaci.

Toto „zbytečné“ kopírování dat navíc velmi negativně ovlivňuje propustnost sítě. Všechny koncové stanice propojené pomocí hubu tvoří tzv. kolizní doménu. Pokud chce daná stanice vysílat na sdíleném médiu, musí nejdříve naslouchat, je-li médium volné – tedy zda na něm právě nevysílá jiná stanice. V důsledku konečné rychlosti šíření elektromagnetické vlny v médiu však může nastat situace, že stanice začne vysílat krátký okamžik po tom, co začala vysílat jiná stanice a dojde tak ke kolizi. O vzniku této kolize jsou stanice informovány speciálním signálem, po kterém počkají náhodnou dobu, než začnou znovu vysílat. Oproti tomu v sítích, které používají switche, tvoří kolizní doménu vždy jeden port switche a zařízení k němu připojené. Tato metoda se označuje jako mikrosegmentace a zabraňuje vzniku kolizí. Moderní switche navíc podporují řadu pokročilých funkcí, jako je tvorba virtuálních lokálních sítí Virtual Local Area Network (VLAN), prioritizaci daného typu provozu a v neposlední řadě umožňují zabránit neoprávněným uživatelům v připojení do sítě.

Směrovač (Router) Směrovač je síťové zařízení, jehož úkolem je směrovat provoz mezi vícero sítěmi. Oproti přepínači pracuje pouze s IP adresami, jedná se tedy o L3 zařízení. Router se většinou nachází na okraji lokální sítě a zpracovává data, která jsou určena k odeslání mimo lokální síť, nebo naopak do ní. Router bývá na odchozí straně připojen k jednomu nebo více dalším routerům. Zde je důležité zmínit, že každý port routeru má vlastní IP adresu a porty sousedních routerů, kterými jsou propojeny, se musí nacházet ve stejné podsíti neboli subnetu.

IP adresu můžeme rozdělit na část síťovou a hostovskou. Všechny stanice (angl. hosts) v rámci jedné sítě mají společnou síťovou část IP adresy. Aby bylo možné určit, kde je hranice mezi těmito dvěma částmi adresy, používá se síťová maska (angl. network mask). Ta je stejně jako IP adresa tvořena čtyřmi oktety, tedy 32 bity. Provedením bitové operace AND mezi IP adresou a síťovou maskou dostaneme adresu sítě:

IP adresa 192.168.1.100:	11000000.10101000.00000001.01100100
Síťová maska 255.255.255.0:	11111111.11111111.11111111.00000000
Adresa sítě 192.168.1.0:	11000000.10101000.00000001.00000000

Právě na základě síťové adresy router rozhodne, kam daný paket vyslat. Spadá-li cílová adresa paketu do definovaného adresního rozsahu, je paket odeslán na odpovídající další zařízení. V případě, že router nemá konkrétně definovaný rozsah, do kterého daná adresa spadá, využije výchozího záznamu, který se označuje jako výchozí brána. Tento proces se nazývá směrování a údaje podle kterých se router rozhoduje jsou obsaženy ve směrovací tabulce. Ta může být nastavená staticky administrátorem sítě, nebo dynamicky pomocí směrovacích protokolů jako je RIP, OSPF, EIGRP, BGP a dalších.

2.2 Možnosti konfigurace síťových zařízení

Dnešní síťové prvky nabízejí řadu cest, kterými je možné přistupovat k jejich nastavení. Níže se pokusím nastínit nejběžnější postupy, včetně jejich případných výhod a nevýhod:

- Konzole
- Telnet
- SSH
- API
- Web GUI

- Vendor-Specific Tool
- Automatizační nástroje

Konzole: Systémová konzole je sice původně fyzické zařízení s klávesnicí a displejem, avšak dnes se již spíše využívají virtuální konzole a terminálové emulátory. Jedná se o základní přístup ke konfiguraci zařízení prostřednictvím CLI. Pro konfiguraci prvku skrze konzoli je potřeba mít fyzické spojení s tímto prvkem. Běžným standardem pro konzolový přístup je komunikační rozhraní RS-232 (sériová linka). Jedná se o bezkolizní rozhraní čistě na fyzické vrstvě RM ISO/OSI. Některé aktivní prvky dnes již poskytují možnost jejich konfigurace přes rozhraní USB, avšak RS-232 je stále mnohem rozšířenější. Rozhraní RS-232 u aktivních prvků většinou využívá konektoru RJ-45. Ačkoliv poslední varianta sériové linky (RS-232C) pochází z roku 1969, často se dnes při konfiguraci aktivních prvků bez tohoto rozhraní neobejdeme, neboť u zařízení v továrním stavu představuje mnohdy jedinou možnost pro konfiguraci, která nám umožní alespoň nastavit další možnosti přístupu, které využívají vyšších vrstev RM ISO/OSI, například Telnet či SSH.

Telnet: Zkratka z *Telecommunication Network* - představuje základní způsob pro vzdálené připojení ke CLI aktivního prvku. Na rozdíl od konzole nevyžaduje přímé fyzické spojení s konfigurovaným zařízením, neboť využívá TCP/IP protokolu. Na druhou stranu je však potřeba, aby zařízení mělo nastavenou IP adresu a povolený přístup prostřednictvím protokolu Telnet. Protokol Telnet typicky využívá portu TCP 23. Značnou nevýhodou tohoto protokolu je však skutečnost, že veškerá komunikace mezi aktivním prvkem a virtuálním terminálem probíhá nešifrovaně. Vzniká tak riziko možného odposlechu této komunikace a získání citlivých informací, které mohou později vést ke kompromitování síťového zařízení. Z tohoto důvodu se protokol Telnet pro vzdálenou správu nedoporučuje.

SSH: *Secure Shell* - zabezpečený komunikační protokol, náhrada za Telnet. Stejně jako Telnet umožňuje vzdálené připojení ke CLI, avšak s tím rozdílem, že zabezpečuje autentizaci obou zařízení a veškerá komunikace je šifrovaná. Existují dvě verze SSH, SSH-1 a SSH-2, přičemž se doporučuje používat druhou verzi, která poskytuje vyšší úroveň zabezpečení. Další výhodou SSH je také možnost autentizace pomocí klíčů, což ještě snižuje možnost neoprávněného přístupu k zařízení. Protokol SSH je v praxi nejrozšířenejší způsob pro přístup k vzdáleným zařízením a jeho druhá verze byla v roce 2006 navržena jako internetový standard. Služba SSH je typicky provozována na portu TCP 22.

API: *Application Programming Interface* - rozhraní pro programování aplikací. Jedná se o sbírku procedur, funkcí, tříd či protokolů, které může programátor využívat. API určuje, jakým způsobem jsou funkce knihovny volány ze zdrojového kódu programu. [6]. Rozhraní API tak umožňuje tvorbu vlastních softwarových řešení pro komunikaci se zařízeními za

účelem získání informací a úpravy jejich konfigurace.

Web GUI: *Web Graphical User Interface* - představuje ideální možnost konfigurace pro méně zkušené uživatele, jedná se o grafické prostředí běžící ve webovém prohlížeči. Toto prostředí však často poskytuje pouze zlomek všech poskytovaných funkcionalit. Tohoto způsobu konfigurace je často využíváno u Small Office / Home Office (SOHO) zařízení. V prostředí větších podniků však může představovat bezpečnostní riziko. Běžný uživatel dokáže mnohem snáze pozměnit konfiguraci síťového zařízení prostřednictvím grafického prostředí, než prostřednictvím CLI. Z tohoto důvodu se nedoporučuje provozovat Web GUI na prvcích síťové infrastruktury.

Vendor-Specific Tool: Výrobci síťových zařízení ve snaze usnadnit správu těchto zařízení poskytují nejrůznější softwarové utility. Jejich účelem je usnadnit nastavení a vyhnout se často zdlouhavé konfiguraci prostřednictvím CLI. Často se také snaží usnadnit prvotní nastavení formou průvodců, což umožní i méně zdatným uživatelům uvést zařízení do základního provozuschopného stavu. Některé pokročilejší nástroje také umožňují správu většího počtu zařízení najednou a poskytují informace o jejich aktuálním stavu.

Automatizační nástroje: Existuje poměrně široká řada jak komerčních tak open-sourcových aplikací pro správu síťové infrastruktury. Tyto aplikace většinou podporují zařízení mnoha výrobců a pro přístup k těmto zařízením používají převážně SNMP či SSH. Tyto programy zároveň slouží jako log-servery a většinou nabízejí grafické prostředí poskytující administrátorovi sítě přehled o stavu jednotlivých zařízení, včetně možnosti upozornění na nejrůznější události. Tyto nástroje však vyžadují jisté nastavení aktivních prvků, proto je nelze využít pro prvotní nastavení při deploymentu zařízení a nacházejí tak své místo spíše v již provozované síti.

2.3 Ansible

2.3.1 Co je Ansible

Ansible je opensource softwarová platforma pro konfiguraci a management (především) serverů, poskytující široké možnosti pro cloud provisioning, správu konfigurací, deployment aplikací či orchestraci služeb. Od základů je navržen pro hromadnou správu mnoha zařízení, s čímž také souvisí jeho state-oriented zaměření. To znamená, že Ansible spíše popisuje chování IT infrastruktury a souvislosti mezi prvky a službami, než aby konfiguroval jeden systém po druhém. V porovnání s konkurenčními nástroji, jako je například Chef nebo Puppet, je Ansible poměrně snadno uchopitelný i pro administrátory začínající se snahou o částečnou automatizaci některých aspektů dané infrastruktury. Jeho jednoduchosti a snadnému nasa-

zení napomáhá také to, že je push-oriented. Na rozdíl od ostatních konfiguračních nástrojů nevyžaduje žádnou instalaci agenta na koncových zařízeních. Pro připojení ke koncovým zařízením se tak nejčastěji používá Secure Shell (SSH). V terminologii Ansiblu se tato zařízení označují jako uzly (angl. *node*) [7]. Informace o jednotlivých uzlech jsou uvedeny v inventáři (angl. *Inventory*), kde je možné definovat parametry jednotlivých uzlů či skupin těchto uzlů.

Ansible je do značné míry komunitním projektem a takřka každý se tedy může podílet na jeho dalším vývoji. Veškeré zdrojové kódy nejnovější verze Ansiblu jsou dostupné na portálu GitHub [8]. Ansible je také dostupný prostřednictvím balíčkovacích služeb. V současné době nelze nainstalovat Ansible na operačním systému Windows. Základní pokyny pro instalaci Ansiblu lze nalézt v dokumentaci dostupné on-line [9].

2.3.2 Historie Ansiblu

Prvotní verze Ansiblu byla vydána 20. února 2012. Jeho autorem je Michael DeHaan, autor softwaru Cobbler, který automatizuje síťově-založené instalace mnoha operačních systémů z centrálního bodu s využitím služeb jako DHCP, TFTP a DNS. Původní společnost Ansible, Inc., která komerčně podporovala a sponzorovala Ansible byla 16. října 2015 převzata společností Red Hat Inc. Od verze 1.7 umožňuje Ansible konfigurací uzlů se systémem Windows. V době psaní této práce je nejnovější stabilní verzí Ansiblu verze 2.0.2.0. Mezi základní cíle projektu Ansible patří minimalistická povaha, konzistence, bezpečnost, vysoká spolehlivost a nízká učící křivka.

2.3.3 Jak Ansible funguje

Ansible je ve své podstatě sadou skriptů a knihoven napsaných v jazyce Python. Provádění samotných konfiguračních příkazů mají na starost právě tyto skripty, tzv. moduly. Ty mají definovaný seznam argumentů, na jejichž základě provedou požadovanou operaci. Argumenty určují požadovaný stav nastavovaných hodnot, změny se tak provedou pouze v případě, že aktuální hodnota neodpovídá té požadované. Opakované spuštění stejného příkazu tak nemá za následek opětovné provádění stejných změn na koncovém uzlu. Většina základních modulů, které jsou součástí instalace Ansiblu je však zaměřena na konfiguraci serverů a vyžaduje podporu jazyka Python na každém uzlu. Ansible pak na základě zvolených argumentů vytvoří skript a spustí jej na zvoleném uzlu. Z tohoto důvodu zatím Ansible neumožňuje konfiguraci většiny síťových prvků. Pro jejich konfiguraci je tak třeba vytvoření modulů, které se spouštějí pouze na straně Ansible serveru a uzlům pak předávají už pouze low-level příkazy. V terminologii Ansiblu se typ tohoto připojení označuje jako lokální (angl. *local*). Potom je možné zvolit libovolný typ připojení podporovaný síťovým prvkem (Telnet, SSH, API,...) a přizpůsobit mu daný modul.

Ansible umožňuje spouštění modulů ve dvou základních režimech. V režimu ad-hoc [10] se vždy spustí pouze jeden zvolený modul s danými parametry 1. Ve druhém režimu se postupně spustí libovolný počet modulů (reprezentovaných příkazy) ze seznamu. Tyto seznamy se označují jako scénáře (angl. *Playbooks*) [11]. Scénáře jsou psány v jazyce Yet Another Markup Language (YAML), díky čemuž jsou velmi přehledné a jednoduché na pochopení. Každý scénář obsahuje název skupiny uzlů, proti kterým budou příkazy použity. Příklad scénáře: 2. Scénáře také umožňují práci s proměnnými, což značně usnadňuje použití modulů proti velkému počtu koncových uzlů. Tyto proměnné lze definovat přímo ve scénáři, nebo v inventáři pro jednotlivé uzly či jejich skupiny.

Inventář je soubor obsahující seznam všech uzlů. Tyto uzly mohou být rozděleny do skupin. Ansible využívá dva druhy inventářů: statické a dynamické. Statický inventář je prostý textový soubor obsahující seznam uzlů a informace o nich. Dynamický inventář je skript napsaný v libovolném jazyce. Jeho jedinou podmínkou je, že při jeho zavolání s parametrem `--list` vrátí do standardního výstupu seznam uzlů s případnými proměnnými. Tento přístup je velmi praktický zejména v případech, že se jedná o velké skupiny uzlů.

```
ansible webservers -m yum -a "name=acme state=latest"
```

Kód 1: Příklad ad-hoc příkazu

Výše uvedený příkaz zavolá nad skupinou uzlů *webservers* modul *yum*, tedy modul ovládající balíčkovací službu distribucí jako Fedora nebo Red Hat Enterprise Linux. Tomuto modulu předá parametry *name=acme* a *state=latest*. I uživateli se základní znalostí linuxových systémů je jasné, co tento příkaz udělá: Na všech serverech ze skupiny *webservers* zkontroluje, že balíček *acme* je nainstalován v nejnovější verzi a pokud není, nainstaluje nejnovější dostupnou verzi.

```
---
- hosts: webservers
  vars:
    http_port: 80
    max_clients: 200
  remote_user: root
  tasks:
  - name: Ensure apache is at the latest version
    yum: name=httpd state=latest
  - name: Write the apache config file
    template: src=/srv/httpd.j2 dest=/etc/httpd.conf
    notify:
    - restart apache
  - name: Ensure apache is running (and enable it at boot)
    service: name=httpd state=started enabled=yes
```

```
handlers:
- name: restart apache
  service: name=httpd state=restarted
```

Kód 2: Příklad scénáře

Výše uvedený kód představuje příklad scénáře pro Ansible. Na začátku souboru je třeba uvést, proti kterým uzlům budou následné příkazy spuštěny, v tomto případě se jedná o skupinu *webservers*. Dále jsou zde zadány hodnoty některých vestavěných proměnných, jako například port, na kterém je provozován HTTP server či jméno uživatele, pod kterým se má Ansible přihlásit ke vzdálenému uzlu. Prvním úkolem v tomto scénáři je zkontrolovat, že Apache web server je nainstalován v poslední verzi, případně tuto verzi nainstalovat. V dalším kroku se přepíše konfigurační soubor Apache serveru tak, aby odpovídal předem připravené šabloně uložené na Ansible serveru. Poté je zavolán handler, který restartuje web server. V posledním kroku se ověří, že služba web serveru je spuštěna a že bude automaticky spuštěna při startu systému. Scénář se spustí příkazem `ansible-playbook playbook.yml`.

2.4 MikroTik

Společnost MikroTik byla založena v roce 1996 v Litvě a zabývá se vývojem a prodejem síťových prvků. Aktivní prvky od společnosti MikroTik využívají operačního systému RouterOS založeného na linuxovém jádře [12]. Zařízení s tímto operačním systémem nabízejí široké možnosti konfigurace, velké množství podporovaných funkcí a relativně nízkou pořizovací cenu v porovnání se zařízeními konkurenčních značek. Z těchto důvodů jsou velmi oblíbené u lokálních ISP pro realizaci přístupové části sítě, především pak pro bezdrátová řešení. Příjmy společnosti MikroTik v roce 2014 činily 135,4 milionů Euro, z čehož 34,4 milionů činí čistý zisk společnosti.

Hlavním produktem společnosti je již zmíněný RouterOS. Tento operační systém poskytuje širokou řadu funkcí, které lze nahrávat prostřednictvím balíčků. Existuje 6 úrovní licencí pro RouterOS. Tyto licence jsou časově neomezené a jsou součástí fyzického zařízení. Licenci je tedy nutné samostatně zakoupit pouze v případě, že chceme instalovat RouterOS na běžném počítači či serveru s architekturou x86. V době psaní této práce existuje již 6. verze tohoto operačního systému. Společnost však také vyvíjí vlastní hardware pod označením RouterBOARD. Série RouterBOARD spolu se systémem RouterOS je zaměřena především na malé a střední poskytovatele bezdrátového internetu.

2.4.1 Možnosti konfigurace zařízení MikroTik

Zařízení RouterBOARD nabízejí tyto možnosti konfigurace:

- Konzole
- Telnet
- SSH
- API
- Web GUI
- WinBox

Přístup prostřednictvím konzole, Telnetu či SSH nabízí standardní CLI, která poskytuje možnost nápovědy či automatické doplňování příkazů. Díky intuitivnosti tohoto rozhraní si na něj lze velmi rychle zvyknout.

WinBox: Jednoduchá utilita pro systémy Windows. Zobrazuje grafické prostředí umožňující konfiguraci zařízení. Jednotlivé sekce nastavení jsou co možná nejblíže stromu konzolových příkazů. Jedná se o proprietární řešení společnosti MikroTik a ve výchozím stavu je pro připojení na straně zařízení použit port TCP 8921 [13]. WinBox navíc umožňuje nalézt dostupná zařízení v síti s využitím proprietárního protokolu MikroTik Neighbor Discovery Protokol (MNDP), což je obdoba známějšího Cisco Discovery Protocol (CDP), který je na zařízeních také podporován. Za zmínku také stojí možnost přenášení souborů mezi PC a zařízením MikroTik, díky čemuž lze snadno nahrávat softwarové balíčky či zálohovat konfiguraci zařízení.

API: Zařízení s RouterOS nabízejí také možnost konfigurace prostřednictvím API. Tato možnost je dostupná od 3. verze RouterOS. Komunikace se zařízením probíhá vysíláním vět, přičemž v odpovědi na jednu větu je přijata jedna nebo více vět. Věty se skládají ze slov a jsou ukončeny slovem nulové délky. Prvním slovem věty je příkaz (*command word*) následovaný atributy (*attribute word*) v libovolném pořadí [14]. Příkaz v sobě zahrnuje cestu ve stromu příkazů a měl by začínat znakem `/`. Příkazy striktně následují CLI. Jednoduchý příkaz pro provedení restartu zařízení bude vypadat takto: `/system/reboot`. Příkaz s atributem pro přidání IP adresy na daném rozhraní bude vypadat takto:

```
/ip/address/add=address=192.168.1.1/24=interface=ether1=disabled=false
```

Po zaslání každé věty vrátí zařízení alespoň jednu strukturovanou odpověď, která buď informuje o úspěšném provedení příkazu, nebo o vzniklé chybě. První slovo odpovědi začíná znakem `!`. Ve výchozím nastavení je pro přístup prostřednictvím API použit port TCP 8728.

MAC-Telnet: Jedná se o proprietární protokol společnosti MikroTik. Jedná se v podstatě o klasický Telnet, avšak s tím rozdílem, že celá komunikace probíhá pouze na L2. To umožňuje komunikovat i se zařízením, které nemá nastavenou IP adresu a nelze k němu tudíž přistoupit některým ze standardních protokolů.

2.5 Základní požadavky na konfiguraci

V rámci této práce se zaměřím na konfiguraci pouze základních parametrů, což umožní provozovat zařízení RouterBOARD v síti. Zde je výčet těchto parametrů:

- Nastavení IP adresy
- Nastavení DHCP serveru
- Nastavení Network Address Translation (NAT)
- Nastavení statických a dynamických cest (Routing)
- Firewall

Nastavení IP adresy: Nastavení IP adresy zařízení představuje základní předpoklad pro provoz síťového zařízení. Zařízení RouterBOARD jsou výlučně routery či L3 switche (samozřejmě s výjimkou bezdrátových Access Pointů), tudíž jsou všechny porty konfigurovatelné na L3 RM ISO/OSI a lze každému z nich přiřadit IP adresu. V rámci této práce se budu věnovat konfigurování pouze v rámci protokolu IP verze 4. Aby byla možná prvotní konfigurace zařízení jinak než sériovou linkou, je ve výchozím nastavení nastavena adresa virtuálního síťového mostu spojujícího všechny porty (kromě prvního). Tato adresa je 192.168.88.1/24. Na prvním ethernetovém portu je u některých modelů spuštěn DHCP klient.

DHCP Server: DHCP server usnadňuje konfiguraci klientských zařízení, na kterých není po připojení třeba nastavovat statickou IP adresu z daného rozsahu. Tuto adresu obdrží zařízení automaticky právě díky DHCP serveru. Ve výchozím nastavení je u většiny modelů spuštěn DHCP server na Local Area Network (LAN) portech. IP adresy nabízené serverem jsou ze stejného rozsahu jako je výchozí IP adresa, tedy 192.168.88.0/24.

Nastavení NAT: NAT je způsob úpravy síťového provozu, který umožňuje překlad několika IP adres na jedinou adresu (případně více). Díky tomuto překladu, který je realizován přepisem informací obsažených v hlavičce IP paketu, není potřeba aby každé zařízení mělo celosvětově unikátní IP adresu. Celou lokální síť s de facto libovolným počtem koncových zařízení lze vnější síti prezentovat jako jedinou adresu. Tento druh překladu síťových adres se označuje jako Source-NAT, tedy překlad na základě zdrojové adresy. Router zapisuje údaje

do NAT tabulky, díky které je pak na základě čísel portů schopen určit, kterému zařízení v lokální síti jsou příchozí pakety určeny. Porty jsou v případě Source-NATu přidělovány dynamicky. Lze definovat i konkrétní porty staticky a určit, na kterou z lokálních adres mají být pakety přicházející na tomto portu přeposílány. Tento způsob překladu se označuje jako Destination-NAT a umožňuje například provozovat Web server přístupný z internetu uvnitř lokální sítě, bez nutnosti adresovat ho veřejnou (celosvětově unikátní) IP adresou. Ve výchozím nastavení je na zařízeních RouterBOARD nastaven Source-NAT typu Masquerade. Jedná se o typ Source-NATu, který umožňuje dynamickou změnu adresy vnějšího (odchozího) portu. Veškerá komunikace z lokální sítě směřující mimo tuto síť je tak přeložena na adresu Wide Area Network (WAN) portu, tedy prvního ethernetového portu.

Nastavení statických a dynamických cest (Routing): Bez záznamů ve směrovací tabulce routeru není možné komunikovat mimo rámec lokální sítě (stejného subnetu). Chceme-li, aby koncová zařízení měla přístup do internetu, je třeba ve směrovací tabulce zavést výchozí záznam, tzv. výchozí bránu (*Default Gateway*) . Tento výchozí záznam se skládá ze speciální IP adresy 0.0.0.0/0 a adresy next-hopu, tedy adresy dalšího zařízení ve vnější síti (routeru), který přepošle vyslané pakety dále. Adresa výchozí brány je většinou distribuována pomocí DHCP. Administrátor má samozřejmě možnost definovat další, specifičtější záznamy ve směrovací tabulce, které popisují jakou cestou má router přistupovat do definované sítě. Tyto záznamy se označují jako statické. Za účelem zvýšení dostupnosti počítačové sítě však bylo navrženo mnoho dynamických směrovacích protokolů. Díky těmto protokolům jsou sousedící routery schopné poskytovat informace o přilehlých sítích a na základě těchto informací si každý router sestaví ve svojí směrovací tabulce záznam, popisující nejlepší cestu do dané sítě. Tyto informace jsou vysílaný většinou v definovaných časových intervalech nebo v případě změny. Zásluhou těchto protokolů je pak síť odolnější například proti poruše některého prvku, protože ostatní směrovače jsou schopny nalézt náhradní cestu do požadované sítě (samozřejmě v případě že taková cesta existuje). Dalším typem záznamů v routovací tabulce jsou vedle statických a dynamických také tzv. Directly Connected, tedy přímo připojené sítě. Router je schopen z IP adresy a síťové masky určit síť, do které tato adresa spadá a záznam o této síti zavede do směrovací tabulky. Tento záznam není možné odstranit. Ve výchozím nastavení je výchozí směrovací záznam přidán pouze je-li z DHCP serveru vnější sítě, do níž je připojen první port, získána adresa výchozí brány.

Firewall: Firewall je filtr síťové komunikace, který na základě definovaných pravidel provádí příslušné akce. Mezi základní akce patří především povolení či zakázání určitého typu komunikace. Firewall implementovaný v RouterOS poskytuje velmi široké možnosti definování pravidel a příslušných akcí. Ve výchozím nastavení jsou nastavena taková pravidla, která umožňují posílat pakety pouze směrem ven z lokální sítě a přijímat pouze takové pakety, které jsou součástí komunikace iniciované lokálním zařízením.

Výchozí konfigurace, která je obsažena ve většině zařízení RouterBOARD při prvním použití, nebo po resetu zařízení tak umožňuje základní síťový provoz. Laicky řečeno, do zařízení stačí ve většině případů připojit příslušné kabely a uživatel bude mít přístup k internetu. Tato konfigurace však představuje pouze nezbytný základ a měla by být upravena dle konkrétních potřeb v souvislosti s topologií sítě.

2.6 Shrnutí a vyhodnocení poznatků

V rámci této kapitoly jsem se snažil nastínit možné přístupy k uchopení problematiky automatizované konfigurace zařízení RouterBOARD s využitím nástroje Ansible. Pro přístup k zařízením by připadalo v úvahu využití protokolu SSH či API, neboť ostatní možnosti přístupu nedávají prostor jejich případnému automatizování pro využití s více zařízeními. Vzhledem k tomu, že Ansible vyžaduje na koncových zařízeních podporu jazyka Python, je bohužel většina existujících modulů k řešení mého problému nepoužitelná. Z tohoto důvodu je tedy nezbytné, vytvořit moduly spouštěné pouze lokálně, které budou samy realizovat přístup k zařízení MikroTik. Za tímto účelem by samozřejmě bylo možné využít existující knihovny pro komunikaci v rámci protokolu SSH, avšak vzhledem ke skutečnosti, že příkazová řádka MikroTiku je navržena primárně pro lidskou interakci, neposkytuje vhodně strukturovaný výstup, který by bylo možné v rámci modulů dobře uchopit. Z tohoto důvodu jsem se tedy pro realizaci rozhodl využít API, které bylo navrženo právě za účelem programovatelně uchopitelné konfigurace a poskytuje strukturovaný výstup.

Při volbě programovacího jazyka pro moduly byl Python mojí první volbou. Ansible sice umožňuje, aby byly moduly psány v libovolném jazyce a klade nároky pouze na jejich správně strukturovaný výstup, avšak vzhledem k tomu, že sám Ansible je psán v jazyce Python, poskytuje pro tento jazyk mnoho knihoven a procedur, které usnadňují tvorbu dalších modulů.

3 Realizace

V této kapitole se budu věnovat detailnímu popisu mnou navrženého řešení včetně konkrétních příkladů použití. Představím zde základní funkce a moduly a příklady jejich použití v rámci Ansiblu. Na začátek bych však rád poskytnul širší pohled na celkové řešení. Mé řešení předpokládá tvorbu modulů pro Ansible, přičemž každý modul je zaměřen na jedno specifické odvětví konfigurace zařízení s RouterOS. Kupříkladu jeden modul bude určen pro nastavení IP adresy na konkrétním portu, jiný modul bude umožňovat nastavení statického

směrování, další modul bude nastavovat DHCP server a tak dále. Primárním úkolem takovýchto modulů je především sestavit ze zadaných parametrů příkazy vhodné pro konfiguraci zařízení a následně poskytnou korektně strukturovaný výstup pro Ansible. Kromě těchto specificky zaměřených modulů je však také třeba vytvoření jedné nebo více funkcí, které budou obstarávat samotnou komunikaci se zařízením a provádět příkazy získané z jednotlivých modulů. Tyto funkce budou tedy působit jako jakási spojná vrstva mezi konfigurovanými zařízeními a jednotlivými moduly, přičemž budou do zařízení posílat konfigurační příkazy získané z modulů a naopak předávat výstup ze zařízení modulům.

3.1 MikroTik API

Již v minulé kapitole bylo zmíněno, že pro přístup ke konfiguraci síťových zařízení je třeba navrhnout vlastní způsob připojení, v tomto případě připojení skrze API. Dokumentace k API na zařízeních s RouterOS je poměrně rozsáhlá a dobře popisuje veškeré požadavky a vlastnosti tohoto rozhraní [14]. Navíc je zde přímo dostupná knihovna pro komunikaci skrze API v jazyce Python. Na serveru GitHub je dostupná vylepšená verze této knihovny [15], jejímž autorem je Piotr Skamruk, vystupující pod pseudonymem *jellonek*. Tato knihovna poskytuje funkce pro připojení k zařízení MikroTik, sestavování a posílání řídících vět a přijímání a zpracování odpovědí. Tuto knihovnu jsem použil v rámci své práce jako základní kámen pro komunikaci s koncovými zařízeními.

3.2 Základní funkce

Funkce a třídy popsané v této části představují sdílené části celkového řešení. Jejich úkolem je provádět úkony společné pro všechny (nebo některé) moduly. Potřebné funkce jsou naimportovány do příslušných modulů. Ansible vyžaduje, aby tyto sdílené moduly byly umístěny ve složce `module_utils`.

RosCore: Obsahuje třídu *Core*, jejímž autorem je již dříve zmíněný Piotr Skamur. Obsahuje funkce pro komunikaci s RouterOS zařízeními a strukturalizaci odpovědí.

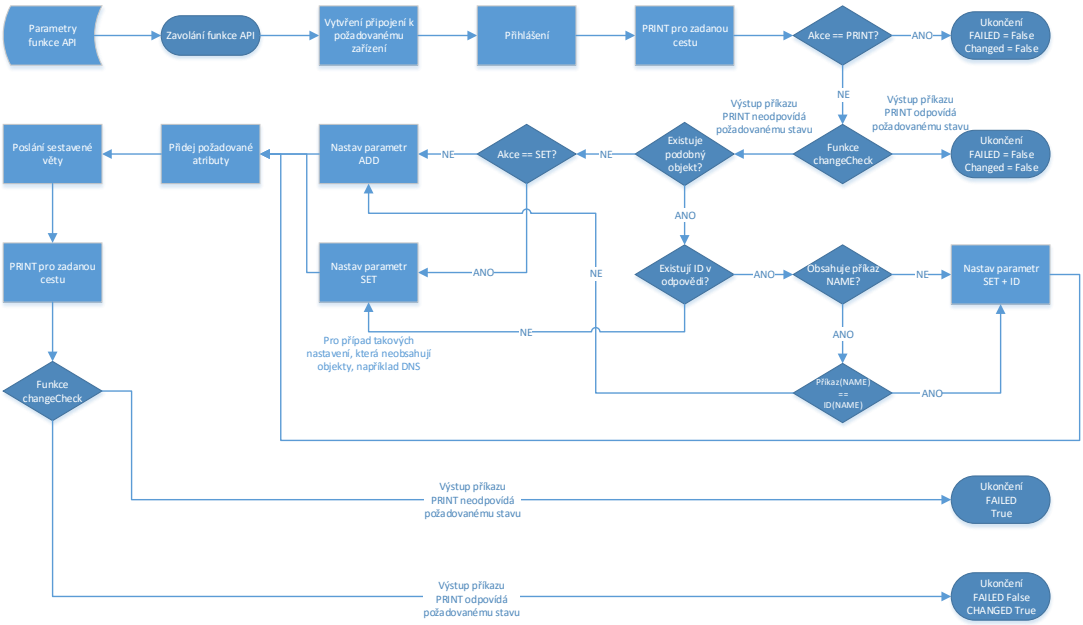
RosAPI: V tomto souboru jsou obsažené některé ze základních sdílených funkcí, z nichž nejdůležitější je funkce je *API* Funkce *API* poskytuje logiku nezbytnou k naplnění požadavku na každý modul Ansiblu, idempotenci. To znamená, že změny jsou provedeny pouze pokud současný stav neodpovídá stavu požadovanému. Funkce *API* přijímá následující parametry:

- `path` - Představuje cestu ve stromu příkazů konfigurace RouterOS. Tato cesta vždy začíná i končí znaménkem `/`. Struktura cest kopíruje strukturu známou z prostředí

CLI, pouze mezery jsou nahrazeny znakem `/`. Kupříkladu cesta vedoucí k nastavení ethernetových portů bude vypadat takto: `/interface/ethernet/`

- `action` - Představuje požadovaný příkaz. Spolu s parametrem `path` tvoří tzv. *command-word*. Mezi typické příkazy patří například: `print`, `add`, `set`, `remove`, `enable`, `disable`. Přidáním příkazu `print` k cestě uvedené v předchozím bodu tak získáme příkaz pro výpis všech existujících ethernetových portů: `/interface/ethernet/print`
- `hostname` - Představuje jméno koncového zařízení nebo jeho IP adresu. Tento parametr tedy udává, ke kterému zařízení se hodláme připojit.
- `username` - Uživatelské jméno nezbytné pro přihlášení do zařízení. Není-li poskytnuto, je použita výchozí hodnota *admin*.
- `password` - Uživatelské heslo nezbytné pro přihlášení do zařízení. Není-li poskytnuto, je použita výchozí hodnota, kterou je prázdné heslo.
- `command` - Tímto parametrem je asociativní pole, jehož klíče představují požadované atributy nastavení a hodnoty jsou požadované hodnoty těchto atributů. Jako příklad vezměme sadu atributů potřebných pro nastavení IP adresy na zvoleném portu zařízení. Pro takový případ může parametr `command` vypadat například takto: {`"address": "192.168.1.1/24", "interface": "ether1", "disabled": "false"`}.
- `port` - Tento parametr představuje TCP port, na kterém je provozováno API. Výchozí hodnota tohoto parametru je *8728*. N tomto portu je na zařízeních MikroTik ve výchozím stavu spuštěna služba API.
- `DEBUG` - Hodnotou tohoto parametru jsou booleovské hodnoty *True* a *False*. Tento parametr slouží pouze k testovacím účelům a odstranění případných chyb poskytnutím dodatečného výstupu. Parametr nelze použít při spouštění modulu pomocí Ansiblu, neboť generovaný výstup nemá požadovanou strukturu.

V prvním kroku porovná funkce *API* současný stav zařízení s požadovaným stavem pomocí příkazu `print` pro danou cestu a funkce *changeCheck*. Odpovídá-li současný stav konfigurace požadovanému, není potřeba provádět žádné změny a odpovídající výstup je vrácen příslušnému modulu. Funkce *changeCheck* má zároveň za úkol určit, zda-li existuje objekt, který má nejvíce shodných atributů s příkazem. Například bude-li příkaz požadovat nastavení IP adresy na portu *ether1*, funkce *changeCheck* určí, zda-li již na daném portu není nastavena IP adresa. V případě, že na portu *ether1* je již nastavena jiná adresa, je třeba provést její změnu příkazem `set` s parametrem *ID* pro daný objekt (port *ether1*). Pokud není nalezen podobný objekt (v tomto případě objekt obsahující atribut `interface=ether1`), bude adresa na příslušný port nastavena pomocí příkazu `add`. Vývojový diagram funkce API je uveden na obrázku 1.



Obrázek 1: RosAPI Flowchart

RosRaw: Tento sdílený modul přijímá stejné parametry jako modul RosAPI, avšak na rozdíl od tohoto modulu neobsahuje logiku pro kontrolu současného stavu konfigurace zařízení. Tento modul slouží pouze pro vyslání přesně definovaných příkazů do zařízení a předává příslušný výstup nadřazenému modulu, který pak na základě vlastní logiky provede další operace. Parametr `command` v případě tohoto modulu není asociativním polem, ale pouze seznamem konkrétních, přesně definovaných atributů.

3.3 Moduly Ansible

3.3.1 Společné znaky Ansible modulů

Ansible klade nároky především na korektně strukturovaný výstup modulu. Tento výstup musí být ve formátu JavaScript Object Notation (JSON). Modul musí poskytnou Ansiblu informace o tom, zda požadovaná akce proběhla v pořádku, a zda došlo ke změnám v konfiguraci zařízení či nikoliv. Za tímto účelem musí vrátit alespoň dvě proměnné `failed` a `changed`, jejichž hodnotny jsou *True* či *False*. Tvorba modulů v jazyce Python je usnadněna tím, že Ansible nabízí procedury pro zpracování vstupních parametrů a pro vrácení korektně strukturovaného výstupu. Každý modul psaný v Pythonu tak potřebuje nejdříve naimportovat sadu základních funkcí příkazem:

```
from ansible.module_utils.basic import *
```

Dále každý modul obsahuje seznam podporovaných parametrů. Jedná se o asociativní pole, v němž jsou uvedeny jednotlivé parametry a jejich možnosti. Ansible například umožňuje nastavení výchozí hodnoty parametru, aliasy parametrů a podobně. Je také například možné definovat, které parametry je nezbytné zadat, případně jaké další parametry je třeba zadat v souvislosti s konkrétním parametrem. Na ukázce níže 3 je uvedena část kódu pocházející z modulu pro nastavení IP adresy. Ukázka neobsahuje samotnou logiku modulu, která volá se zadanými parametry funkci *API* a na základě odpovědi předá Ansiblu příslušný výstup.

```
#!/usr/bin/python

# Import Ansible module parameters
ansible = AnsibleModule(
  argument_spec=dict(
        hostname=dict(required=False, type='str', aliases=["host"]),
        username=dict(required=False, type='str', aliases=["user"]),
        password=dict(required=False, type='str', aliases=["pass"]),
        port=dict(required=False, type='int', default=8728),
        address=dict(required=False, type='str', aliases=['ip', 'addr']),
        interface=dict(required=True, type='str', aliases=['int']),
        disabled=dict(required=False, type='str', choices=['true', 'false'])

        ),
supports_check_mode=False,
)

def main():
        from ansible.module_utils.RosAPI import API
        from ansible.module_utils.RosAPI import intCheck
        # Initialize phase

        # Required parameters
        hostname = ansible.params['hostname']
        username = ansible.params['username']
        password = ansible.params['password']
        port = ansible.params['port']
        path = "/ip/address/"
        action = ""
        command = {"address": ansible.params['address'],\
                   "interface": ansible.params['interface']}
        #
        # Module logic ommited
        #
from ansible.module_utils.basic import *

if __name__ == '__main__':
```

```
main()
```

Kód 3: Ukázka modulu v jazyce Python

3.3.2 Popis jednotlivých modulů

Seznam podporovaných parametrů je vždy uveden na začátku zdrojového kódu modulu. V této sekci proto jen vysvětlím, k čemu který modul slouží. Parametry modulů se snaží co nejpřesněji kopírovat příkazy známé z CLI, což by mělo zkušenějšímu uživateli usnadnit práci s nimi. Pomlčky v názvech konfiguračních parametrů jsou nahrazovány podtržítky, jelikož syntaxe jazyka YAML nepovoluje pomlčky u klíčových hodnot.

mt_ip: Tento modul slouží pro nastavení IP adresy na zvoleném portu zařízení. Modul nejdříve provede kontrolu, zda-li vybraný port existuje a není-li zadaná IP adresa již nastavena na jiném portu. Poté provede nastavení IP adresy.

mt_dhcp_net: Jeden z celkem tří modulů pro konfiguraci DHCP serveru. Tento modul má na starost vytvoření sítě pro DHCP server.

mt_ip_pool: Další modul pro konfiguraci DHCP serveru, jehož úkolem je vytváření adresního poolu pro server.

mt_dhcp_srv: Poslední modul pro konfiguraci DHCP serveru, jehož úkolem je především nastavit informace poskytované serverem, tedy například adresu výchozí brány, Domain Name System (DNS) serveru, NTP serveru a podobně. Jedním z parametrů tohoto modulu je port, na kterém má být server provozován.

mt_dns: Jednoduchý modul pro nastavení parametrů DNS serveru na zařízení. Umožňuje nastavit adresy dalších serverů a ukládání záznamů v lokální paměti.

mt_fetch: Tento modul zprostředkovává interakci s funkcí *fetch*. Tato funkce umožňuje stahování či nahrávání datových souborů přímo ze zařízení. S využitím tohoto modulu je možné přistupovat k FTP či HTTP serveru, například za účelem stažení aktualizačních balíčků či nahrání záloh konfigurace. Pomocí tohoto modulu je tak možné například stahovat složité konfigurační skripty z FTP serveru.

mt_static_route: Modul pro přidávání statických směrovacích záznamů. Logickými parametry jsou proto adresa cílové sítě a adresa next-hopu. Tento modul tak slouží například pro nastavení výchozí brány.

mt_nat: Modul pro správu nastavení překladu síťových adres. Umožňuje nastavení překladu na základě zdrojové adresy (Source-NAT) či cílové adresy (Destination-NAT). Pomocí tohoto

modulu je tak například možné nastavit zdrojový NAT typu *masquerade* na portu směřujícím do veřejné sítě, či nastavit přesměrování příchozí komunikace na zařízení v místní síti.

mt_firewall: Modul spravující nastavení pravidel firewallu. Poskytuje pouze základní možnosti nastavení s využitím odchozích a příchozích portů zařízení, zdrojových a cílových adres a TCP/UDP portů, či stavu spojení.

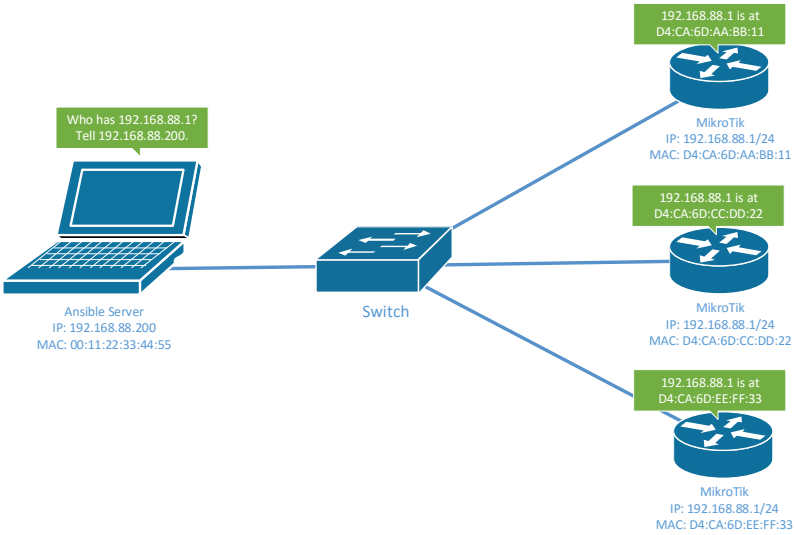
mt_rip: Modul umožňující nastavení dynamického směrovacího protokolu Router Information Protocol (RIP). Pomocí tohoto modulu je možné nastavit, které sítě se mají propagovat a skrze které porty zařízení. Kromě konkrétních sítí je také možné nastavit propagaci přímo připojených sítí, statických směrovacích záznamů či záznamů získaných prostřednictvím dalších dynamických směrovacích protokolů.

mt_brctl: Modul pro konfiguraci síťových mostů. Vytváření síťových mostů umožňuje spojit více portů zařízení do jednoho logického portu. Tohoto modulu lze využít například při vytváření virtuálních lokálních sítí. Ty totiž na zařízeních MikroTik vyžadují vytvoření individuálních mostů, do kterých jsou poté přidány vybrané porty.

mr_raw: Tento modul poskytuje možnost vyslání libovolného příkazu do zařízení. Vyžaduje přesnou syntaxi požadovaného příkazu a neobsahuje žádnou logiku pro kontrolu a zpracování odpovědí. Slouží především pro jednoduché úkony, jako například restart zařízení, spuštění skriptu či přidání uživatelského účtu.

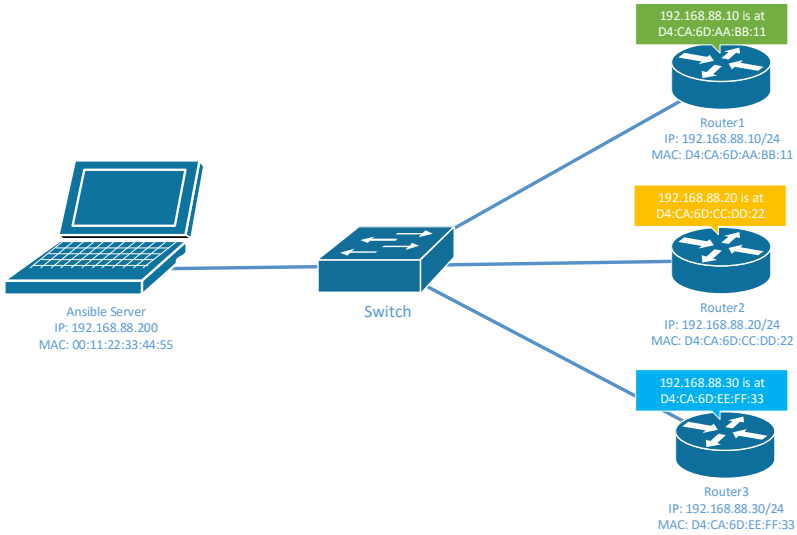
3.4 Prvotní konfigurace

Jedním z požadavků při realizaci problematiky této práce je také navržení postupu pro konfiguraci zařízení, která se nacházejí ve výchozím továrním nastavení. Konfigurace v tomto stavu je popsána v kapitole 2.5. V důsledku tohoto nastavení, není možné přistoupit ke konfiguraci většího počtu zařízení MikroTik přímo pomocí *API*, především proto, že všechna zařízení mají nastavenou stejnou IP adresu, 192.168.88.1/24. V důsledku toho není možné jednoznačně určit MAC adresu pojící se k adrese 192.168.88.1/24, což je nezbytný předpoklad pro komunikaci na L2. Komunikace mezi zařízeními na L3 se v takovém případě stává zcela nepředvídatelnou. Situace je demonstrována na obrázku 2. Address Resolution Protocol (ARP) dotaz ze stanice s Ansible serverem vyvolá odpovědi od všech tří routerů.



Obrázek 2: Konflikt IP adres

Zařízení MikroTik naštěstí umožňují konfiguraci i bez nastavené IP adresy prostřednictvím proprietárního protokolu MAC-Telnet. V našem případě je tedy možné nastavit na zařízeních unikátní IP adresu pomocí MAC-Telnetu, což umožní další přístup skrze API. Klient pro MAC-Telnet je dostupný na serveru GitHub a jeho autorem je Håkon Nessjøen [16]. Bohužel se mi však nepodařilo vytvořit modul pro Ansible v jazyce Python, který by dokázal v rámci volání podprocesů využít MAC-Telnet pro komunikaci se zařízeními. Byl jsem proto nucen ustoupit k jedinému funkčnímu řešení, které využívá jednoduchého *expect* skriptu a na základě zadaných parametrů provede pouze nejzákladnější příkazy. Jedná se o skripty `mtbash_clean`, který vynutí restart zařízení bez nahrání výchozí konfigurace a `mtbash_default`, který na základě vstupních parametrů nastaví IP adresu na zvoleném portu, nastaví jméno zařízení a vytvoří nový uživatelský účet. Situaci nyní popisuje obrázek 3.



Obrázek 3: Situace po konfiguraci prostřednictvím MAC-Telnetu

3.5 Inventář

Inventář Ansiblu je ve své podstatě seznamem všech známých uzlů. Typicky se jedná o soubor se jménem `hosts`. Uvnitř tohoto souboru mohou být uzly rozdělené do skupin, případně zde lze uvést libovolné parametry jednotlivých uzlů či skupin. Je však dobrým zvykem ukládat údaje o uzlech do samostatných souborů, které jsou umístěny ve složce `host_vars`, přičemž jméno každého souboru je jménem daného uzlu. Obdobně lze zapsat parametry týkající se skupiny uzlů do souboru, jehož jménem je název skupiny a tento soubor umístit do složky `group_vars`. Příklad souboru `hosts` pro situaci z obrázku 3 může vypadat takto:

```
[routers]
Router1 IP=192.168.88.10
Router2 IP=192.168.88.20
Router3 IP=192.168.88.30

[routers:vars]
username=admin
password=""
api_port=8728
```

Kód 4: Příklad statického inventáře s parametry

Inventář tvoří skupina *routers* se třemi uzly, *Router1*, *Router2* a *Router3*, z nichž každý má jako jeden z parametrů uvedenou svoji IP adresu. Dále je zde vidět sekce se skupinovými parametry. Všechny uzly používají pro přihlášení výchozí jméno a heslo.

3.5.1 Dynamický inventář

Ansible je velmi robustní nástroj navržený pro správu stovek až tisíců koncových uzlů. Spravovat statický inventář ručně pro takovéto počty uzlů by bylo nemožné. Proto Ansible nabízí využití tzv. dynamického inventáře (*dynamic inventory*), což je skript, který nějakým způsobem získá informace o uzlech. Opět platí, že tento skript může být napsán v libovolném jazyce, pouze musí vracet strukturovaný JSON formát do standardního výstupu. Tento výstup je pak převzat Ansiblem a použit jako inventář.

Pro usnadnění konfigurace většího počtu zařízení MikroTik jsem vytvořil skript pro generování dynamického inventáře. Skript používá k nalezení zařízení v síti informace z CDP protokolu a tyto informace následně předá Ansiblu. Jelikož jméno každého uzlu musí být unikátní, nelze vždy použít jako jméno uzlu jméno nastavené v zařízení. To platí především v případě, že se snažíme nakonfigurovat více zařízení, která se nacházejí v továrním nastavení. V takovém případě použije skript jako jméno uzlu zdrojovou MAC adresu. Příklad výstupu skriptu `cdp_discovery` pro situaci z obrázku 2 bude vypadat takto:


```
{
  "_meta":{
        "hostvars":{
                "d4:ca:6d:aa:bb:11":{
                        "ID":"MikroTik",
                        "IP":"192.168.88.1",
                        "Platform":"MikroTik",
                        "Port":"ether1",
                        "Source_MAC":"d4:ca:6d:aa:bb:11",
                        "Version":"6.32.3"
                },
                "d4:ca:6d:cc:dd:22":{
                        "ID":"MikroTik",
                        "IP":"192.168.88.1",
                        "Platform":"MikroTik",
                        "Port":"ether1",
                        "Source_MAC":"d4:ca:6d:cc:dd:22",
                        "Version":"6.32.3"
                },
                "d4:ca:6d:ee:ff:33":{
                "ID":"MikroTik",
                "IP":"192.168.88.1",
                "Platform":"MikroTik",
                "Port":"ether1",
                "Source_MAC":"d4:ca:6d:ee:ff:33",
                "Version":"6.32.3"
                },
        }
```

```
},
"mikrotiks":{
        "hosts":[
        "d4:ca:6d:aa:bb:11",
        "d4:ca:6d:cc:dd:22",
        "d4:ca:6d:ee:ff:33"
        ],
        "vars":{
                "default_password":"",
                "default_user":"admin"
        }
}
}
```

Kód 5: Příklad výstupu skriptu pro dynamický inventář

Z výstupu je patrné, že všechna nalezená zařízení se stala součástí skupiny `mikrotiks`. Objekt `_meta` slouží pro předání informací o jednotlivých uzlech. Skript navíc přidává výchozí jméno a heslo do skupinových proměnných. Aby nebylo nutné spouštět skript při každé změně konfigurace, lze jej využít jako základní kámen pro vytvoření statického inventáře. Stačí pouze předat strukturovaný výstup z příkladu 5 dalšímu skriptu, který tento výstup použije k vytvoření statického inventáře. Autorem tohoto skriptu je David Wittman a skript je volně dostupný na serveru GitHub [17]. Vytvoření statického inventáře navíc umožňuje provádět změny jednotlivých parametrů a použít tyto parametry jako argumenty pro některé moduly v rámci playbooku.

3.6 Ansible Playbook

Scénáře Ansiblu poskytují široké možnosti strukturalizace. Aby bylo snadnější udržet scénáře přehledné, je možné rozdělit do více souborů a tyto sady poté používat v rámci nadřazeného scénáře. Za tímto účelem je možné vytvořit tzv. role, které poskytují sady příkazů pro provedení určitých změn. Role jsou reprezentovány složkami ve složce *roles*. Jako první příklad vezměme roli *init*, která má za úkol smazat tovární konfiguraci v zařízení a následně nastavit IP adresu, změnit jméno zařízení a vytvořit nového uživatele. Role *init* bude tvořena třemi soubory: `clear.yml`, `default.yml` a `main.yml`. Soubor `clear.yml` bude obsahovat příkazy pro odstranění tovární konfigurace. V tomto případě se jedná o jediný příkaz, který lokálně spustí *expect* script pro komunikaci skrze MAC-Telnet. Výrazy ve dvojitých složených závorkách představují proměnné :

```
---
- name: Clear MikroTik Configuration
```

```
    local_action: script mtbash_clean.sh {{Source_MAC}} {{default_user}} {{
        default_password}}
```

Kód 6: Obsah souboru `clear.yml`

Dále soubor `default.yml` bude obsahovat příkaz pro nastavení IP adresy na daném portu, jména zařízení a vytvoření nového uživatele:

```
---
  - name: Add Default Configuration
    local_action: script mtbash_default.sh {{Source_MAC}} {{default_user}} {{
        default_password}} {{ID}} {{new_user}} {{new_password}} {{Port}} {{IP}}
```

Kód 7: Obsah souboru `default.yml`

Soubor `main.yml` musí být obsažen v příslušné roli, neboť slouží jako výchozí soubor použitý při volání dané role. V našem případě tedy soubor spojuje dva předešlé pomocí příkazu `include`. Aby bylo možné odstranit tovární nastavení, je třeba zařízení restartovat. Poté je potřeba počkat, než bude možné posílat další příkazy. K tomu slouží příkaz `pause`:

```
---
  - include: clear.yml
  - pause: minutes=1
  - include: default.yml
```

Kód 8: Obsah souboru `main.yml`

Nyní se dostáváme k "opravdovému"playbooku, který spustí roli *init*. Na začátku scénáře je uvedeno jeho jméno, typ připojení, definované proměnné a samozřejmě skupina uzlů, vůči které budou příkazy spouštěny. Scénář je uložen v souboru `site.yml` uloženého v kořenové složce:

```
- name: Init Playbook # This playbook resets MikroTik devices and assingns IP
    addresses to connected ports
  hosts: mikrotiks
  connection: local
  gather_facts: no
  vars:
    default_user: admin
    default_password: "''"
    new_user: ansible
    new_password: ansible
  roles:
  - init
```

Kód 9: Obsah souboru `site.yml`

```
root@linux:/ansible-mikrotik# ansible-playbook -i hosts site.yml --fork=1

PLAY [Init Playbook]
    ****************************************************

TASK [init : include]
    ****************************************************
included: ./roles/init/tasks/clear.yml for Router1, Router2

TASK [init : Clear MikroTik Configuration]
    ***********************************
changed: [Router1 ->localhost]
changed: [Router2 ->localhost]

TASK [init : pause]
    ************************************************************
Pausing for 60 seconds
(ctrl+C then 'C' = continue early, ctrl+C then 'A' = abort)
ok: [Router1]

TASK [init : include]
    ****************************************************
included: ./roles/init/tasks/default.yml for Router1, Router2

TASK [init : Add Default Configuration]
    ***************************************
changed: [Router1 ->localhost]
changed: [Router2 ->localhost]

PLAY RECAP
    ********************************************************************
Router1                    : ok=5    changed=2    unreachable=0    failed=0
Router2                    : ok=4    changed=2    unreachable=0    failed=0
```

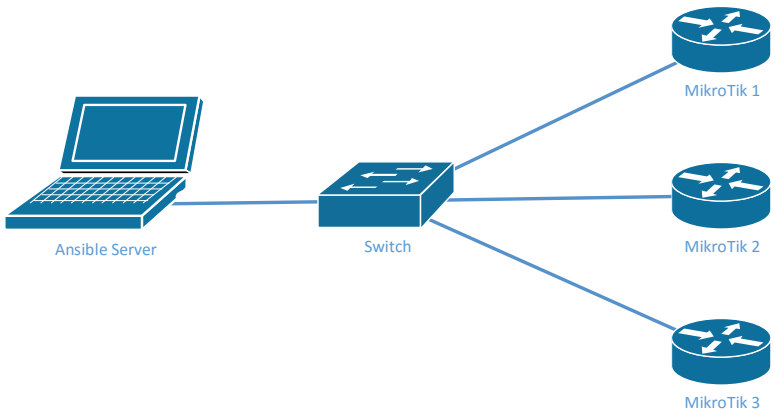
Kód 10: Výstup Ansiblu pro roli *init*

Scénáře se spouštějí příkazem `ansible-playbook`, přepínač `-i` určuje soubor inventáře. Parametr `--fork=1` je v případě spouštění příkazů využívajících MAC-Telnet nezbytný. Parametr určuje kolik paralelních spojení může Ansible vytvořit, avšak spuštění více instancí MAC-Telnetu může vést k chybám. Ve výstupu je patrný postup jednotlivých úkolů, na závěr je uveden přehledný souhrn celé hry.

3.7 Příklady použití

3.7.1 Testovací pracoviště

Při vývoji a testování modulů jsem využíval virtuálních routerů MikroTik. Na stránkách výrobce jsem volně dostupné obrazy ISO instalačních CD či přímo před-připravené instalace pro využití ve VMware či Hyper-V [18]. Jako fyzické zařízení pro testování jsem používal RouterBOARD model RB2011UiAS-2HnD-IN [19] uvedený na obrázku 5. V průběhu této práce jsem na testovaných zařízeních provozoval RouterOS ve verzích *6.32.3*, *6.34.4* a *6.35.2*.



Obrázek 4: Příklad zapojení testovací laboratoře

Obrázek 5: RB2011UiAS-2HnD-IN

3.7.2 Příklad 1

V tomto příkladu uvedu ukázku scénáře, jehož úkolem je nastavit DNS server, DHCP server, IP adresu na dalším portu, statickou cestu do zvolené sítě, Source-NAT typu masquerade pro

první port a Destination-NAT pro vzdálený přístup k místnímu serveru.

```
- name: Mikrotik TEST1
  hosts: mikrotiks
  connection: local
  gather_facts: no
  vars:
    dhcp_pool: testpool
    username: ansible
    password: ansible

  tasks:
  - name: DNS Setup
    mt_dns: username={{username}} hostname={{IP}} password={{password}} servers
        =192.168.116.1,8.8.8.8 remote_requests="true"
  - name: IP address setup
    mt_ip: username={{username}} hostname={{IP}} password={{password}} address
        ={{et2addr}} interface=ether2 disabled="false"
  - name: DHCP Pool Setup
    mt_ip_pool: username={{username}} hostname={{IP}} password={{password}}
        pool_name={{dhcp_pool}} pool_range=192.168.116.10-192.168.116.20
  - name: DHCP Server Setup
    mt_dhcp_srv: username={{username}} hostname={{IP}} password={{password}} name
        =TestServer address_pool={{dhcp_pool}} disabled="true" interface=ether1
  - name: DHCP Options Setup
    mt_dhcp_net: username={{username}} hostname={{IP}} password={{password}}
        network_address=192.168.116.0/24 gateway=192.168.116.1 dns_server
        =192.168.116.1
  - name: Set static route
    mt_static_route: username={{username}} hostname={{IP}} password={{password}}
        dst_address=192.168.10.0/24 gateway=192.168.10.1
  - name: Source NAT
    mt_nat: username={{username}} hostname={{IP}} password={{password}}
        out_interface=ether1 chain=srcnat action=masquerade
  - name: Port Forwarding to machine SSH
    mt_nat: username={{username}} hostname={{IP}} password={{password}} chain=
        dstnat action=dst-nat dst_port=8022 to_addresses=192.168.116.20
        to_ports=22 protocol=tcp
```

Kód 11: Scénář *MikroTik TEST1*

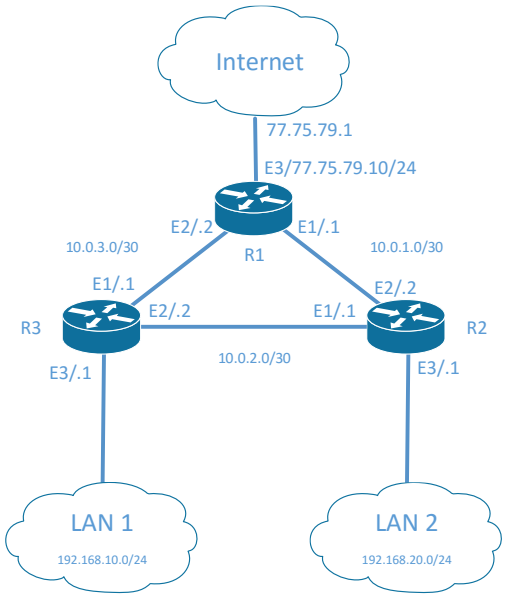
```
root@linux:/ansible-mikrotik# ansible-playbook -i hosts test_playbook.yml

PLAY [Mikrotik TEST1]
    ************************************************************

TASK [DNS Setup]
```

```
    ************************************************************
changed: [Router2]
changed: [Router1]

TASK [IP address setup]
    ******************************************************
changed: [Router2]
changed: [Router1]

TASK [DHCP Pool Setup]
    ******************************************************
changed: [Router1]
changed: [Router2]

TASK [DHCP Server Setup]
    ******************************************************
changed: [Router2]
changed: [Router1]

TASK [DHCP Options Setup]
    ******************************************************
changed: [Router1]
changed: [Router2]

TASK [Set static route]
    ******************************************************
changed: [Router1]
changed: [Router2]

TASK [Source NAT]
    ************************************************************
changed: [Router1]
changed: [Router2]

TASK [Port Forwarding to machine SSH]
    ***************************************
changed: [Router2]
changed: [Router1]

PLAY RECAP
    ********************************************************************
Router1                    : ok=8    changed=7    unreachable=0    failed=0
Router2                    : ok=8    changed=7    unreachable=0    failed=0
```

Kód 12: Výstup Ansiblu pro scénář *MikroTik TEST1*

3.8 Příklad 2

V tomto příkladu budeme modelovat topologii sítě z obrázku 6. Postupně nastavíme IP adresy na všech označených portech a poté zprovozníme protokol RIP mezi routery. Dále na routerech 2 a 3 nastavíme DHCP server pro sítě LAN1 a LAN2. V posledním kroku nastavíme na Routeru 1 statický záznam pro výchozí bránu a na veřejném portu spustíme zdrojový překlad adres. Informace o výchozí bráně bude propagována pomocí protokolu RIP i mezi routery 2 a 3.



Obrázek 6: Topologie malé podnikové sítě

Pro prvotní konfiguraci předpokládejme topologii z obrázku 4, přičemž pro připojení k zařízením používáme port *ether4*. Všechny tyto porty mají spolu s kontrolním PC adresy z rozsahu *192.168.100.0/24*. Nejdříve je v inventáři potřeba definovat všechny potřebné hodnoty:

```
---
Router1
ID: Router1
IP: 192.168.100.10/24
et1addr: 10.0.1.1/30
et2addr: 10.0.3.0/30
et3addr: 77.75.79.10/24
---
Router2
ID: Router2
```

```
IP: 192.168.100.20/24
et1addr: 10.0.1.2/30
et2addr: 10.0.2.1/30
et3addr: 192.168.20.1/24
pool_range: "192.168.20.10 - 192.168.20.200"
network: 192.168.20.0/24
---
Router3
ID: Router3
IP: 192.168.100.30/24
et1addr: 10.0.3.1/30
et2addr: 10.0.2.2/30
et3addr: 192.168.10.1/24
pool_range: "192.168.10.10 - 192.168.10.200"
network: 192.168.10.0/24
```

Kód 13: Inventář pro topologii z obrázku 6

```
---
  - name: All routers
    hosts: Router1,Router2,Router3
    connection: local
    gather_facts: no
    vars:
      username: ansible
      password: ansible
    tasks:
     - name: Ethernet 1 IP address
       mt_ip: username={{username}} hostname={{IP}} password={{password}}
         interface=ether1
         address={{et1addr}}
         disabled="false"
     - name: Ethernet 2 IP address
       mt_ip: username={{username}} hostname={{IP}} password={{password}}
         interface=ether2
         address={{et2addr}}
         disabled="false"
     - name: Ethernet 3 IP address
       mt_ip: username={{username}} hostname={{IP}} password={{password}}
                interface=ether3
                address={{et3addr}}
                disabled="false"
       - name: RIP setup
         mt_rip: username={{username}} hostname={{IP}} password={{password}}
                interfaces=ether1,ether2
                redistribute_connected="true"
                distribute_default="if-installed
                send=v2
```

```
                receive=v2
                authentication=md5
                authentication-key=P@ssw0rd123
---
  - name: Access routers
    hosts: Router2,Router3
    connection: local
    gather_facts: no
    vars:
      username: ansible
      password: ansible
    tasks:
    - name: IP Pool
      mt_ip_pool: username={{username}} hostname={{IP}} password={{password}}
        pool_name=pool
        pool_range={{pool_range}}
    - name: DHCP Server Setup
      mt_dhcp_srv: username={{username}} hostname={{IP}} password={{password}}
        name=dhcp-local
        address_pool=pool
        disabled="false"
        interface=ether3
    - name: DHCP Options Setup
      mt_dhcp_net: username={{username}} hostname={{IP}} password={{password}}
        network\_address={{network}}
        gateway={{et3addr}}
        dns_server=8.8.8.8
---
  - name: Border router
    hosts: Router1
    connection: local
    gather_facts: no
    vars:
      username: ansible
      password: ansible
    tasks:
    - name: Static IP address
      mt_static_ip: username={{username}} hostname={{IP}} password={{password}}
        destination_address="0.0.0.0/0"
        gateway=77.75.79.1
    - name: Source-NAT
      mt_nat: username={{username}} hostname={{IP}} password={{password}}
        chain=srcnat
        out_interface=ether3
        action=masquerade
```

Kód 14: Scénáře pro topologii z obrázku 6

4 Závěr

Toto téma bakalářské práce jsem si zvolil především z důvodu zájmu o problematiku automatizované konfigurace sítí. Se zařízeními RouterBOARD jsem se setkal již dříve, avšak jejich konfiguraci jsem realizoval vždy pouze s využitím příkazové řádky nebo grafického WinBoxu. V tomto ohledu mi zpracování problematiky této práce značně rozšířilo obzory. Především seznámení s nástrojem Ansible osobně hodnotím jako velmi prospěšné. Ačkoliv se mi zprvu zdál ne příliš vhodný ke konfiguraci síťových zařízení bez podpory jazyka Python, ve výsledku se ukázal být nedocenitelným nástrojem už jen pro správu inventářů a široké možnosti organizace a strukturalizace příkazů pomocí scénářů a rolí. Jedná se opravdu o velmi robustní a mocný nástroj, ale přesto se jeho základní principy dají pochopit a naučit vcelku rychle. Navíc díky možnosti psát vlastní moduly v libovolném jazyce může kdokoliv se základní znalostí programování přispívat a dále rozšiřovat už tak široké schopnosti tohoto nástroje. Tím se pomalu dostávám k další zkušenosti, kterou jsem získal v rámci tohoto projektu. Touto zkušeností bylo seznámení se s jazykem Python. Programování jsem se nikdy příliš nevěnoval a musím přiznat, že jsem poměrně nedávno žil v domnění, že při studiu telekomunikačních sítí nebude znalost programování nezbytná. Žijeme však v době, kdy se poměrně dlouho zaběhnutý systém správy informačních technologií přesouvá od konzolí do virtualizovaných cloudů a ony nekonečné údržby a obsluhy těchto systému přebírají orchestrační aplikace a nástroje jako Ansible. Stejně jako u Ansiblu i pro Python platí nízká učící křivka a navíc má poměrně široké možnosti nasazení právě v oblasti správy síťové infrastruktury.

4.1 Naplnění stanovených cílů

Cílem této práce bylo rozšířit schopnosti nástroje Ansible pro konfiguraci síťových zařízení. První překážkou byla skutečnost, že ačkoliv je Ansible nástrojem bez agentů, vyžaduje pro většinu modulů podporu jazyka Python. Po seznámení se s různými možnostmi konfigurace zařízení MikroTik jsem se rozhodl pro realizaci využít API, které slibovalo nejlepší uchopitelnost z pohledu programového přístupu ke konfiguraci. Volba API se ukázala jako dobrá volba díky níž bylo snazší realizovat následné moduly pro Ansible. Původní myšlenka při realizaci modulů byla taková, že moduly budou mít minimální velikost a budou sloužit především k získání adekvátního vstupu a poskytnutí výstupu pro Ansible. Moduly by tak pouze předaly potřebné parametry nadřazené funkci, která by na základě vlastní logiky umožnila provádět libovolné změny v konfiguraci bez ohledu na jejich charakter. Bohužel vzhledem k rozsáhlým možnostem nastavení a jejich vzájemného provázání se mi nepodařilo vytvořit jednu všemocnou funkci, která by zajišťovala provedení změn. Rozhodl jsem se proto přenechat část logiky celého procesu přímo v modulech, ve kterých lze snáze ošetřit

konkrétní případy spojené s konkrétním nastavením. Podařilo se mi tak vytvořit moduly pro nastavení některých základních funkcí zařízení MikroTik. Mezi tyto funkce patří adresace síťových rozhraní, konfigurace DHCP a DNS serveru, konfigurace síťových mostů, správa firewallových pravidel, překlad síťových adres, dynamické routovaní pomocí protokolu RIP. Některých dalších nastavení může být dosaženo pomocí přímých příkazů prostřednictvím modulu *mt_raw*. Kromě tvorby samotných modulů jsem se také věnoval problematice dynamických inventářů pro použití s velkým počtem koncových zařízení a problémům spojeným s prvotní konfigurací zařízení RouterBOARD. Výsledkem mého snažení byl skript generující inventář pro Ansible na základě informací získaných z odposlechu CDP rámců. Problematika prvotní konfigurace se ukázala být poněkud složitější, než jsem očekával. Za účelem prvotní konfigurace samozřejmě existují různé možnosti, většina z nich však vyžaduje hlubší znalost problematiky a nezbytné vybavení. Příkladem takového řešení může být například nahrání vlastního obrazu operačního systému do zařízením prostřednictví bootování ze síťového TFTP serveru. Mojí snahou však bylo řešení, které by kladlo minimální požadavky na přidané prostředky a bylo ovladatelné přímo z Ansiblu. Po konzultaci jsem se rozhodl využít proprietárního protokolu MAC-Telnet, který zařízení MikroTik podporují. Bohužel jsem narazil na problémy s ovládáním MAC-Telnetu prostřednictvím skriptů v jazyce Python a byl jsem nucen vydat se pro mne jedinou funkční cestou, kterou bylo ovládání MAC-Telnetu pomocí expect skriptu. V důsledku toho se možnosti konfigurace prostřednictvím MAC-Telnetu omezili na minimum nezbytné k umožnění přístupu prostřednictvím API. Samozřejmě je zde spousta prostoru, kam by se tato práce dala dále rozvíjet. Příkladem by mohlo být například vytvoření pruginu pro Ansible, který by přímo realizoval spojení se zařízeními MikroTik prostřednictvím API. Nad takovýmto připojením by mohla pracovat řada modulů. Díky takovému přístupu by pak bylo možné vytvořit další typy spojení pro další výrobce síťových zařízení a moduly tak udělat univerzálnější.

Při realizaci jsem čerpal z uvedených zdrojů, především pak z online dokumentace Ansiblu a MikroTiku. Dalšími zdroji pak byly internetová fóra a různé komunitní skupiny spojené s Ansiblem, Pythonem a operačním systémem RouterOS.

Reference

[1] Wikipedia. Information technology infrastructure library — wikipedia, the free encyclopedia. [https://en.wikipedia.org/wiki/ITIL], 2014. [Online].

[2] Dan Conde. Network automation: More than scripting. [http://www.networkcomputing.com/data-centers/network-automation-more-scripting/819125107], 2015. [Online].

[3] Cisco Systems, Inc. Cisco application centric infrastructure. [http://www.cisco.com/c/en/us/solutions/data-center-virtualization/application-centric-infrastructure/index.html]. [Online].

[4] IBM Global Services. Improving systems availability. [http://www.dis.uniroma1.it/~irl/docs/availabilitytutorial.pdf], 1998. [Online].

[5] Richard Harris. Network reliability. [http://seat.massey.ac.nz/143465/Lectures/Network%20Reliability_2_1s.pdf]. [Online].

[6] Wikipedia. Api — wikipedia, the free encyclopedia. [https://cs.wikipedia.org/wiki/API], 2016. [Online].

[7] Lorin Hochstein. *Ansible: Up and Running*. O'Reilly Media, Inc., May 2015.

[8] Ansible. Github - ansible. [https://github.com/ansible/ansible]. [Online].

[9] Ansible. Documentation - ansible. [http://docs.ansible.com/ansible/]. [Online].

[10] Ansible Docs. Introduction to ad-hoc commands. [http://docs.ansible.com/ansible/intro_adhoc.html], 2016. [Online].

[11] Ansible Docs. Intro to playbooks. [http://docs.ansible.com/ansible/playbooks_intro.html], 2016. [Online].

[12] Wikipedia. Mikrotik — wikipedia, the free encyclopedia. [https://en.wikipedia.org/wiki/MikroTik], 2016. [Online].

[13] MikroTik Wiki. Manual:winbox - mikrotik wiki. [http://wiki.mikrotik.com/wiki/Manual:Winbox]. [Online].

[14] MikroTik Wiki. Manual:api - mikrotik wiki. [http://wiki.mikrotik.com/wiki/Manual:API]. [Online].

[15] Piotr Skamruk. Routerboard python API. [https://github.com/jellonek/rosapi]. [Online].

[16] Håkon Nessjøen. Mac-telnet Client. [https://github.com/haakonnessjoen/MAC-Telnet]. [Online].

[17] David Wittman. Ansible Dynamic Inventory Converter. [https://gist.github.com/DavidWittman/189816c1c2b3a25bd1b18f5f9f681262]. [Online].

[18] MikroTik. Mikrotik - Downloads. [http://www.mikrotik.com/download]. [Online].

[19] MikroTik. Mikrotik - RB2011UiAS-2HnD-IN. [http://routerboard.com/RB2011UiAS-2HnD-IN]. [Online].