České vysoké učení technické v Praze
Fakulta elektrotechnická

Katedra telekomunikační techniky
Obor: Komunikační sítě a internet

Softwarově definované sítě v prostředí datového centra

SDN in a datacenter environment

Diplomová práce

Vypracoval: Bc. Marek Rušin
Vedoucí práce: Ing. Ján Kučerák
Rok: 2024

ZADÁNÍ DIPLOMOVÉ PRÁCE

I. OSOBNÍ A STUDIJNÍ ÚDAJE

Příjmení: **Rušin** Jméno: **Marek** Osobní číslo: **483535**

Fakulta/ústav: **Fakulta elektrotechnická**

Zadávající katedra/ústav: **Katedra telekomunikační techniky**

Studijní program: **Elektronika a komunikace**

Specializace: **Komunikační sítě a internet**

II. ÚDAJE K DIPLOMOVÉ PRÁCI

Název diplomové práce:

Softwarově definované sítě v prostředí datového centra

Název diplomové práce anglicky:

SDN in datacenter environment

Pokyny pro vypracování:

Proveďte analýzu a porovnání různých architektur a implementací softwarově definovaných sítí (SDN). Navrhněte a realizujte na základě analýzy prostředí Algotech cloudu testovací SDN řešení, které umožní integraci se stávajícím systémy a případné nasazení ve skutečném produkčním provozu datového centra. Automatizujte proces vytváření nových virtuálních instancí a integrace se související fyzickou infrastrukturou. Zanalyzujte bezpečnost řešení a možnosti škodlivých útoků na SDN.

Seznam doporučené literatury:

[1] HOLÍK, F., NERADOVÁ, S., Softwarově definované sítě, 2019, [cit. 27.12. 2022], ISBN: 978-80-7560-235-0
[2] KREUTZ, D., RAMOS, V., M., F., VERÍSSIMO, E., P., ROTHENBERG, E., CH., AZODOLMOLKY,S., UHLIG, S. Software-Defined Networking: A Comprehensive Survey, [online], Proceedings of the IEEE, 2015, 103(1), 14-76 s, [cit. 10. 2. 2023], DOI 10.1109 / PROC 2014.2371999.
[3] Red Hat OpenStack Platform 10 Networking Guide, [online], Red Hat Inc., 2019

Jméno a pracoviště vedoucí(ho) diplomové práce:

Ing. Ján Kučerák katedra telekomunikační techniky FEL

Jméno a pracoviště druhé(ho) vedoucí(ho) nebo konzultanta(ky) diplomové práce:

Datum zadání diplomové práce: **20.02.2023** Termín odevzdání diplomové práce: _____

Platnost zadání diplomové práce: **16.02.2025**

Ing. Ján Kučerák podpis vedoucí(ho) práce	podpis vedoucí(ho) ústavu/katedry	prof. Mgr. Petr Páta, Ph.D. podpis děkana(ky)

III. PŘEVZETÍ ZADÁNÍ

Diplomant bere na vědomí, že je povinen vypracovat diplomovou práci samostatně, bez cizí pomoci, s výjimkou poskytnutých konzultací. Seznam použité literatury, jiných pramenů a jmen konzultantů je třeba uvést v diplomové práci.

. _____
Datum převzetí zadání

_____
Podpis studenta

Prehlásenie

Prehlasujem, že som predloženú prácu vypracoval samostatne a že som uviedol všetkú použitú literatúru.

V Prahe dňa ....
Marek Rušin

Poďakovanie

Touto cestou by sa chcel poďakovať Ing. Jánovi Kučerákovi za jeho vedenie, rady a trpezlivosť pri vypracovaní diplomovej práce. Tiež by som chcel poďakovať mojej rodine a priateľom za ich podporu počas štúdia.

Abstrakt

Softvérovo definované siete predstavujú spôsob, pomocou ktorého možno siete spravovať centrálne a prostredníctvom softvéru. S mnohými ďalšími výhodami sa preto stávajú súčasťou podnikových sietí či sietí dátových centier. Táto diplomová práca rozoberá základné princípy SDN a popisuje virtualizačné platformy používané firmou Algotech na spravovanie virtuálnej infraštruktúry dátového centra. V praktickej časti diplomovej práce bol vytvorený nástroj, v ktorom administrátor prostredníctvom Jenkins užívateľského rozhrania dokáže spravovať časť virtuálnej infraštruktúry dátového centra. Potrebné kroky pri prenose nastavenia virtuálneho prvku z Jenkins užívateľského rozhrania do cieľovej virtualizačnej platformy sú v rámci riešenia plne automatizované.

Kľúčové slová: SDN, VMware, automatizácia, Jenkins, Python, Terraform

Abstract

Software-defined networks represent how networks can be managed centrally and by software. With many other advantages, they become part of enterprise or data center networks. This thesis discusses the basic principles of SDN and describes the virtualization platforms used by Algotech to manage the virtual infrastructure of the Algotech's data center. In the practical part of the thesis, a tool was developed that allows an administrator to manage a portion of the virtual infrastructure of a data center via the Jenkins user interface. The steps required to transfer the settings of the virtual elements from the Jenkins user interface to the target virtualization platform are fully automated within the solution.

Keywords: SDN, VMware, automation, Jenkins, Python, Terraform

Obsah

Zoznam obrázkov

Kapitola 1

Úvod

Počítačové siete založené na rodine protokolov TCP/IP už od doby svojho vzniku zažívajú konzistentný nárast v podobe veľkosti sietí, objemu prenášaných dát či požiadavok na výkon siete. S príchodom technológií ako 5G, cloud alebo IoT sa taktiež razantne zvyšujú požiadavky kladené na siete, ako napr. vyššia prenosová rýchlosť, nižšia latencia, škálovateľnosť či bezpečnosť.

Správa sietí tradičným prístupom, ktorá zahŕňa ovládanie sieťového prvku pomocou príkazového riadku či konfiguračných súborov, je obmedzujúca pre potreby moderných dátových sietí. Tradičné siete sú statické a nedokážu sa rýchlo prispôsobovať aktuálnym požiadavkám na sieť.

Riešením tohto problému v podnikových sieťach či v sieťach dátových centier je zavedenie automatizácie a obmedzenie priamej interakcie sieťových administrátorov so sieťovou infraštruktúrou. Najnovšie trendy v oblasti moderných počítačových sietí preto napr. zahrňujú [1], [2]:

- **DevOps** - predstavuje spoluprácu medzi vývojom (development) a IT prevádzkou (operations) s cieľom automatizovať nasadzovanie softvéru. DevOps tak pomáha zrýchliť spoločnostiam dodávanie nových aplikácií a služieb [3].

- **Programovateľná infraštruktúra** - programovateľnosť sietí sa snaží docieliť zníženie interakcií medzi človekom a sieťovým zariadením. Automatizácia a skriptovanie sieťových úloh je charakteristickým znakom programovateľnej infraštruktúry.

- **Software-defined networking** - označuje súbor techník, ktoré sa používajú na správu a riadenie počítačových sietí. Vďaka SDN možno programovať sieťové zariadenia buď prostredníctvom kontroléra alebo iného externého mechanizmu. SDN umožňuje spravovať celú sieť ako jeden celok. SDN nie je priamo určitý typ technológie alebo produktu, ale prístup, ktorý v sebe

zahŕňa pojmy ako napr. rozdelenie riadiacej a dátovej roviny sieťového zariadenia, virtualizácia sieťových funkcií, centralizovanie riadenia siete a smerovanie k otvoreným protokolom.

- **Intent-based networking** - je forma spravovania sietí za použitia strojového učenia alebo umelej inteligencie pre automatizáciu údržby a správu sietí. IBN tvorí nadstavbu SDN [4].

Práve SDN, ktorým je venovaná táto diplomová práca, umožňujú skrz centralizované riadenie zjednodušiť správu siete. Nasadením SDN možno docieliť zníženie operačných nákladov, zjednodušiť zavedenie novej služby či pridanie nového prvku do topológie siete [5].

1.1 Ciele práce

Diplomová práca je rozdelená na teoretickú a praktickú časť. Cieľom teoretickej časti je priblížiť problematiku SDN, popísať kľúčové pojmy spojené s SDN (architektúra, nasadenie, bezpečnosť SDN) a porovnať SDN s tradičnou sieťovou architektúrou. Ďalej úlohou teoretickej časti je popísať virtualizačné nástroje používané v dátovom centre Algotechu - vSphere a NSX-T. Hlavnou časťou diplomovej práce je praktická časť, ktorá sa zaoberá implementovaním automatizácie pri spravovaní virtuálnych instancií v prostrediach NSX-T a vSphere.

Kapitola 2

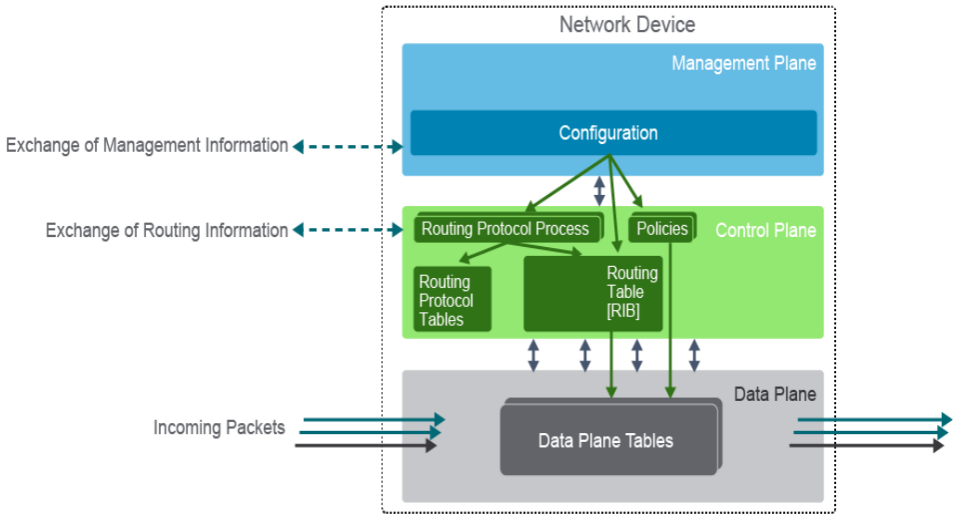
Tradičné počítačové siete

Pred samotným definovaním SDN táto kapitola stručne vysvetľuje pojem tradičných sietí, keďže stále tvoria dôležitú súčasť moderných počítačových sietí. Ďalej kapitola popisuje architektúru sieťových zariadení a uvádza činnosti, ktoré vykonávajú jednotlivé roviny v sieťovom zariadení. Záver kapitoly popisuje nedostatky tradičných sietí.

Úlohou počítačových sietí je prepájať zariadenia v sieti a sprostredkovať komunikáciu medzi koncovými zariadeniami. Vo svojej štruktúre počítačové siete používajú dedikované hardvérové zariadenia (prepínače, smerovače, firewally...), ktoré plnia pevne stanovené funkcie (napr. prepínanie rámcov, smerovanie paketov), a tým zabezbečujú konektivitu v sieti. Pojem tradičných počítačových sietí referuje k prístupu, ako sú tieto zariadenia konfigurované a spravované. Tradičný prístup vyžaduje priamu interakciu administrátora siete so sieťovým zariadením prostredníctvom CLI alebo GUI. Napriek nutnosti priamej ľudskej interakcie so zariadením je stále tento spôsob spravovania sietí aj v dnešnej dobe populárny [6].

2.1 Architektúra tradičných počítačových sietí

V každom sieťovom zariadení prebiehajú procesy, ktoré možno rozdeliť do troch skupín, ktorým sa tiež hovorí logické roviny. Tieto roviny sú: dátová, riadiaca a spravovacia. Logické roviny sú v tradičných sieťach priamo prepojené a obsahuje ich každé sieťové zariadenie. Počas bežnej prevádzky siete pozostáva komunikácia medzi zariadeniami prevažne z paketov dátovej roviny, ktoré sú určené koncovým užívateľom. Sieťové zariadenia sú optimalizované tak, aby tieto pakety dokázali efektívne spracovať a odoslať. Počet paketov určených pre riadiacu a spravovaciu rovinu zariadenia je podstatne menší [1].



Obr. 2.1: Architektúra zariadenia v tradičnej počítačovej sieti [1]

- **Dátová rovina** - primárnou úlohou dátovej roviny je preposielať rámce a pakety k cieľovému zariadeniu. S cieľom čo najrýchlešie preposielať dáta je implementovaná logika v dátovej rovine navrhnutá čo najjednoduchšie, aby správy bolo možné preposielať len za použitia dedikovaného hardvéru. Ten pozostáva zo špecializovaných ASIC (príp. FPGA) čipov a pamäte s rýchlym prístupom TCAM. Údaje, ako spracovať, a kam odoslať dátový tok, sú uložené v dátových štruktúrach známych ako tabuľky. Medzi takéto tabuľky možno zaradiť CAM, FIB a ďalšie. Uloženie týchto tabuliek do TCAM pamäti umožňuje ASIC čipom veľmi rýchlo dohľadať údaje, ako spracovať dáta.

  Typické úlohy, ktoré vykonáva zariadenie v dátovej rovine sú, napr. pridať (odstrániť) 802.1Q hlavičku, vybrať podľa FIB tabuľky, z akého rozhrania odoslať paket s príslušnou cieľovou IP adresou atď. V dátovej rovine možno zároveň implementovať jednoduché filtrovanie dátového toku za použitia ACL [1].

- **Riadiaca rovina** - úlohou riadiacej roviny je vytvárať inštrukcie pre dátovú rovinu, ako spracovať dátový tok. Ku činnostiam riadiacej roviny tiež patrí výmena riadiacich informácií medzi sieťovými zariadeniami. Riadiaca rovina podľa konfigurácie vytvára tabuľky (RIB, ARP tabuľka, CAM tabuľka a pod.), vďaka ktorým môže dátová rovina vykonávať svoju činnosť. Zmenami v týchto tabuľkách riadiaca rovina ovláda prevádzku dátovej roviny. Pakety určené pre riadiacu rovinu sú spracované za použitia CPU zariadenia. Tie

môžu byť pre zariadenie priamo určené (napr. výmena dát medzi smerovacími protokolmi) alebo sú preposlané z dátovej roviny za účelom úpravy dát (napr. inkrementovanie hodnoty TTL v IPv4 hlavičke paketu).

Medzi typické úlohy v riadiacej rovine možno zaradiť činnosti rôznych smerovacích protokolov (napr. OSPF), ARP v IPv4 či NDP v IPv6. [1].

- **Spravovacia rovina** - táto rovina zahrňuje nástroje a protokoly, pomocou ktorých môže sieťový administrátor konfigurovať a monitorovať zariadenie, prípadne zbierať štatistiky zo spracovaného dátového toku. Spravovacia rovina využíva pre spravovanie zariadenia a siete najmä protokoly SSH, SNMP, HTTP a mnohé ďalšie [1].

Funkcionalitu sieťového zariadenia možno tak popísať niekoľkými vetami. Prostredníctvom spravovacej roviny sa vykonáva konfigurácia zariadenia. Následne na základe tejto konfigurácie riadiaca rovina zariadenia vytvára inštrukcie pre dátovú rovinu.

Procesy a protokoly sú v tradičných sieťach implementované tak, že každé zariadenie si vytvára vlastnú instanciu určitého procesu (napr. každý smerovač má iný priebeh OSPF protokolu). Aby mohli procesy v sieti správne fungovať, vymieňajú si sieťové zariadenia informácie o ich priebehoch. Tým, že zariadenia rozhodujú o riadiacich úlohách samostatne, má tradičný prístup spravovania sietí distribuovaný charakter. To so sebou prináša niekoľko výhod, ako napr. zaručenie redundancie v sieti [7].

2.2 Nevýhody tradičných počítačových sietí

V tradičných počítačových sieťach je potrebné pri konfigurácii pristupovať individuálne ku každému sieťovému prvku. S priamou ľudskou interakciou so zariadením, ako aj so samotnou distribuovanou architektúrou tradičných sietí sa viaže niekoľko nevýhod: [8]

- **Decentralizované rozhodovanie** - sieťové prvky rozhodujú samostatne o riadiacich úlohách, ako je napr. smerovanie paketov. V tradičných sieťach neexistuje centrálny prvok, ktorý by dokázal ovládať všetky zariadenia naraz a taktiež býva zložité určiť aktuálny stav siete.
- **Lokálna konfigurácia** - počiatočná konfigurácia sieťových prvkov zvyčajne vyžaduje lokálny prístup. Lokálny spôsob konfigurácie môže byť ale aj vyžadovaný pri odstraňovaní určitých chýb. V prípade geograficky rozsiahlych oblastí to môže znamenať zvýšenie finančných nákladov a väčšiu časovú medzeru pre vykonanie zmien v konfigurácií.

- **Náchylnosť k chybám** - vzhľadom k náročnosti konfigurácie rozsiahlych počítačových sietí dochádza často ku chybám pri konfigurácií. Tieto chyby môžu spôsobiť výpadky siete alebo znížiť jej výkonnosť.
- **Nedeterministická doba opráv** – proces hľadania a opráv konfiguračných chýb je v tradičných počítačových sieťach časovo nepredvídateľný. Navyše určité typy chýb nemožno opraviť za použitia vzdialeného prístupu. Ako už bolo zmieňované, to prináša zvýšenie finančných nákladov a predĺženie doby, kým bude sieť uvedená do požadovaného stavu.
- **Náročná konfigurácia** – problematika správnej konfigurácie tradičných počítačových sietí je rozsiahla a vyžaduje časovo náročné vzdelávanie.
- **Obmedzená škálovateľnosť** - akékoľvek zmeny v sieti vyžadujú statické konfigurácie na všetkých sieťových prvkoch, ktorých sa táto zmena týka. Pri prechode na novú konfiguráciu tak vznikajú krátkodobé výpadky v konfigurovaných častiach siete.
- **Statická konfigurácia** - tradičné siete neumožňujú automaticky meniť konfiguráciu v závislosti na dynamických udalostiach.
- **Závislosť na jednom výrobcovi** - každý výrobca sieťových prvkov zvyčajne používa vo svojich zariadeniach vlastný proprietárny softvér. Sieťové prvky od rôznych výrobcov sú medzi sebou nekompatibilné, a preto počítačové siete zvyčajne pozostávajú zo zariadení od jedného výrobcu.

Tradičná správa počítačových sietí má stále vitálnu rolu pri konfigurovaní a spravovaní sietí. Ale s narastajúcimi požiadavkami na škálovateľnosť a programovateľnosť sietí je tento prístup nevyhovujúci. Preto sa začal klásť dôraz na automatizáciu. Jedným z riešení je napr. použitie nástrojov pre správu konfigurácie ako sú Ansible, Chef alebo Puppet či použitie automatizačných skriptov na konfiguráciu zariadení. Ďalšou možnosťou ako docieliť programovateľnosť a automatizáciu v sieťach je použitie SDN, ktorým je venovaná nasledujúca kapitola.

Kapitola 3

Softvérovo definované siete

Kapitola sa zaoberá opisom kľúčových pojmov spojených s SDN, a to hlavne v oblasti architektúry SDN či sieťových prvkov v SDN. V kapitole je tiež uvedený krátky úvod do protokolu OpenFlow a stručný popis virtualizácie sieťových funkcií (NFV), ktorá sa vzájomne dopĺňa s SDN. Záver kapitoly zhrňuje vlastnosti SDN, popisuje bezpečnostné hrozby v SDN a možnosti nasadenia SDN.

SDN je sada techník na riadenie a správu počítačových sietí, ktorá ponúka dynamické a programové prístupy k monitorovaniu a konfigurácií siete. V SDN je oddelené priame spojenie medzi dátovou a riadiacou rovinou. Sieťové prvky umiestnené v dátovej rovine nevykonávajú svoje pevne dané funkcie, ale prijímajú inštrukcie od externej entity - SDN kontroléra, ako spracovať dátový tok. SDN takisto oddeľuje hardvér od softvéru. Riadiaca rovina sa v rámci SDN presúva do softvéru. Toto centralizované riadenie umožňuje správcom siete programovať a spravovať celú sieť prostredníctvom jedného uzlu. Formálne možno SDN definovať nasledujúcimi štyrmi bodmi [9]:

- Riadiaca a dátová rovina sú od seba oddelené. Zo sieťových zariadení je odstránené riadenie, a tak sa zo zariadení stávajú jednoduché prvky na preposielanie správ.
- Rozhodnutia o preposielaní správ sú založené na tzv. toku (flow). Flow obsahuje sadu filtrov a inštrukcií medzi zdrojovým a cieľovým zariadením. Zovšeobecnením tohto toku je možné zjednotiť správanie zariadení v dátovej rovine ako prepínače, smerovače, firewally atď. Programovateľnosť tokov zaručuje vysokú mieru flexibility.
- Riadiaca rovina je umiestnená do samostatnej entity, tzv. SDN kontroléra alebo NOS (Network Operating System). NOS je softvérová platforma umiestnená na serveri, ktorej úlohou je riadenie zariadení v dátovej rovine za použitia abstraktného pohľadu na celú sieť.

- Sieť je programovateľná pomocou aplikácií umiestnených v aplikačnej rovine. SDN kontrolér následne riadi sieť v súlade s prijatými inštrukciami od aplikačnej roviny.

SDN sa od tradičných sietí líši tým, že ovládanie siete je riešené softvérovo a samotný kontrolér môže priamo komunikovať s aplikáciami prostredníctvom API. Vývojári aplikácií tak môžu programovať sieť priamo, čo umožňuje rýchlo a jednoducho vykonávať akékoľvek zmeny v sieti.

Ďalším pojmom spojeným s SDN je NFV (network function virtualization). Ako napovedá názov tejto technológie, NFV slúži na virtualizáciu sieťových zariadení ako napr. smerovače, firewally a load-balancery, ktoré majú v tradičných sieťach hardvérovú podobu. NFV nahrádza funkcionalitu týchto zariadení virtuálnymi zariadeniami umiestnenými na serveri. Pomocou NFV môžu spoločnosti ľahko nasadzovať nové sieťové služby bez nutnosti zaobstarania dedikovaného hardvéru [10].

NFV a SDN nie sú na sebe závislé, ale majú mnoho podobností. Oba prístupy využívajú virtualizáciu a sieťovú abstrakciu, ale spôsob akým oddeľujú funkcie a abstrahujú zdroje je odlišný. SDN oddeľuje dátovú a riadiacu rovinu siete s cieľom vytvoriť sieť, ktorá je centrálne spravovateľná a programovateľná. NFV abstrahuje sieťové funkcie od dedikovaného hardvéru. NFV podporuje SDN tým, že poskytuje infraštruktúru, ktorú možno integrovať do SDN. Pomocou NFV a SDN možno vytvoriť siete, ktoré sú flexibilné a efektívne využívajú pridelené hardvérové zdroje [10].

3.1 Architektúra SDN

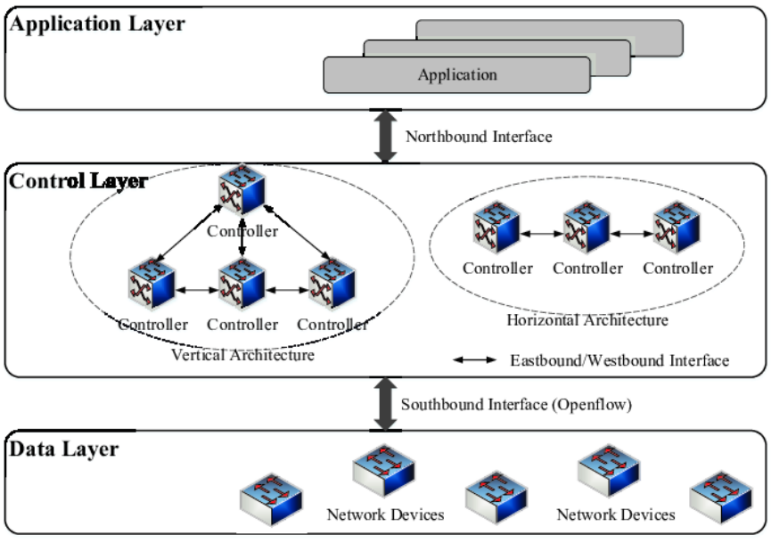
Podobne ako v architektúre tradičných sietí sa aj architektúra SDN delí na tri logické roviny. Logické roviny v SDN sa delia na dátovú, riadiacu a aplikačnú. Každá logická rovina komunikuje len so svojou susediacou rovinou.

3.1.1 Logické roviny SDN

- **Dátová rovina**
  Rovnako ako v architektúre tradičných počítačových sietí pozostáva dátová rovina zo sieťových zariadení určených na preposielanie PDU. Rozdiel oproti dátovej rovine v tradičných sieťach spočíva v tom, že tieto zariadenia sú teraz jednoduchými preposielacími prvkami, ktoré ovláda SDN kontrolér. Úlohou týchto zariadení je prijímať, spracovávať, odosielať, prípadne zahadzovať PDU podľa inštrukcií od kontroléra. V určitých prípadoch možno prijatú správu poslať SDN kontroléru [9].

- **Riadiaca rovina**
  Riadiaca rovina obsahuje jeden alebo viacero SDN kontrolérov, ktoré sa zvyčajne nachádzajú na fyzickom či virtuálnom serveri. Hlavnou úlohou riadiacej roviny je vytvárať inštrukcie pre SDN zariadenia v dátovej rovine a dohliadať na dodržiavanie konfigurácie siete. SDN kontrolér prijíma požiadavky z aplikačnej vrstvy, ktoré následne spracuje a vytvorí sadu inštrukcií pre zariadenia v dátovej rovine. Najmä za učelom monitorovania siete kontrolér tiež môže preposielať určité dáta z dátovej do aplikačnej roviny [11].

- **Aplikačná rovina**
  Obsahuje sadu aplikácií a nástrojov, ktoré slúžia pre konfigurovanie a správu siete, monitoring dátového toku a pod. Príkladmi týchto aplikácii sú napr. load-balancery, aplikačné firewally a aplikácie pre dohľad na QoS. Tradičné siete pre vykonávanie uvedených úloh vyžadujú špecializované sieťové zariadenia. V SDN sú tieto zariadenia nahradené aplikáciami, ktoré ovládajú správanie dátovej roviny prostredníctvom jedného alebo viacerých kontrolérov. Aplikácie prostredníctvom severného rozhrania prenášajú inštrukcie do SDN kontroléra [11].



Obr. 3.1: Architektúra SDN [12]

3.1.2 Komunikačné rozhrania SDN

Narozdiel od tradičných sietí, SDN nepoužíva priame spojenie medzi susednými logickými rovinami. Preto komunikáciu medzi susednými logickými rovinami spro-

stredkovávajú tri komunikačné rozhrania. A to južné (southbound), severné (northbound) a rozhranie západ-východ(west/eastbound).

- **Južné (southbound) rozhranie**
  Úlohou južného rozhrania je sprostredkovať komunikáciu medzi riadiacou a dátovou rovinou, čím je dôležitým nástrojom pre oddelenie týchto dvoch rovín. Južné rozhranie umožňuje SDN kontroléru zisťovať topológiu siete, odosielať pravidlá toku do SDN zariadení a získavať štatistiky z dátového toku od sieťových zariadení. Medzi hlavné výzvy na tomto rozhraní patrí heterogenita sieťových zariadení a podporovaných protokolov. Južné rozhranie používa množstvo protokolov, ale najrozšírenejším priemyselným štandardom južného rozhrania je protokol OpenFlow, ktorému je venovaná samostatná subkapitola. Medzi ďalšie z niektorých štandardov južného rozhrania patria [9], [13]:

  - **OVSDB** (Open vSwitch Database) je typ protokolu južného rozhrania, ktorý je určený pre pokročilú správu prepínačov Open vSwitch. Okrem bežných funkcionalít protokolu Openflow, umožňuje OVSDB napr. riadiacim prvkom vytvárať virtuálne instancie prepínačov, nastavovať QoS na ich rozhraniach alebo vytvárať tunelové rozhrania. OVSDB je doplnkovým protokolom k OpenFlow pre Open vSwitch.
  - **POF** (Protocol Oblivious Forwarding) je jedným z hlavných konkurentov protolu OpenFlow. Oproti OpenFlow, ktorý používa tabuľky tokov pre riadenie SDN zariadení, využíva POF všeobecnú sadu inštrukcií toku s názvom FIS (Flow Instruction Set). V POF je rozbor paketu úlohou kontroléra a SDN zariadenia iba paket spracujú a prepošlú ho ďalej kontroléru.
  - **ForCES** (Forwarding and Control Element Separation) umožňuje flexibilnejší prístup k správe tradičných sietí bez zmeny súčasnej architektúry siete, tzn. bez potreby zavedenia logicky centralizovaného riadenia. Riadiaca a dátová rovina sú oddelené, ale nachádza sa v tom istom sieťovom zariadení. Riadiacu časť sieťového prvku tak možno ovládať prostredníctvom firmvéru tretej strany.

- **Severné (northbound) rozhranie**
  Aplikačná rovina používaja severné rozhranie pre interakciu s riadiacou rovinou. SDN kontroléry podporujú rôzne programovacie jazyky a ponúkajú širokú škálu API pre severné rozhranie. Medzi najpoužívanejšie API patria REST API využívajúce protokol HTTP alebo ad-hoc API. Avšak v súčastnosti neexistuje štandard pre severné rozhranie. S cieľom štandardizovať severné rozhranie, vytvorila ONF (Open Network Foundation) skupinu s

názvom North Bound Interface Working Group (NBI-WG), ktorej úlohou je vytvoriť otvorený štandard pre toto rozhranie. Štandardizovaním severného rozhrania by sa docielila lepšia spolupráca medzi aplikáciami a kontrolérom a zjednodušenie správy siete [9].

- **Rozhranie západ-východ (west/eastbound interface)**
  Ako už bolo zmieňované, zvyčajne jeden SDN kontrolér riadi SDN zariadenia v dátovej rovine. Ak ale sieť narastá, použitie jedného kontroléru nemusí stačiť pre riadenie siete. Preto je v rozsiahlych sieťach potrebné použiť viacero SDN kontrolérov pre zaistenie väčšej škálovateľnosti a spoľahlivosti. V prípade použitia viacerých kontrolérov ovláda zvyčajne každý z nich určitú doménu siete. Výmenu informácií v rámci riadiacej roviny zabezpečuje rozhranie západ-východ. Západné rozhranie sa používa pre komunikáciu medzi SDN kontrolérmi. Východné rozhranie zabezpečuje výmenu informácií medzi riadiacou rovinou SDN a distribuovanou riadiacou rovinou v tradičných sieťach. Oba rozhrania zatiaľ neboli štandardizované, a preto výrobcovia kontrolérov zvyčajne používajú svoje vlastné proprietárne API [13].

3.2 Typy architektúr riadiacej roviny SDN

S príchodom SDN bolo hlavným úsilým využívať jeden, fyzicky centralizovaný, kontrolér. Ale pre zaistenie redundancie a dodatočnej škálovateľnosti v sieti sa začalo preferovať použitie viacerých kontrolérov. V SDN existujú dva typy architektúr riadiacej roviny, a to fyzicky centralizovaná a fyzicky distribuovaná. Logicky distribuovanú architektúru možno následne deliť na ďalšie subkategórie [14].

- **Fyzicky centralizovaná architektúra**
  Fyzicky centralizovaná architektúra riadiacej roviny využíva práve jeden kontrolér pre ovládanie celej siete, ako sa zatiaľ uvažovalo priebežne v práci. Výhodou tohto prístupu je ovládanie celej siete prostredníctvom jedného uzla, ktorý nemusí riešiť komunikáciu s inými kontrolérmi. Napriek tejto výhode je s uvedeným návrhom spojených množstvo nevýhod. Samostatný kontrolér predstavuje tzv. jediný bod zlyhania (single point of failure) v sieti. Pri jeho výpadku alebo narušení jeho bezpečnosti dôjde k ochromeniu a strate kontroly nad sieťou.

- **Fyzicky distribuovaná architektúra**
  Riadaca rovina vo fyzicky distribuovanej architektúre využíva viac kontrolérov (či už fyzických alebo logických) pre ovládanie siete. Kontroléry medzi sebou spolupracujú a zaručujú vyššiu škálovateľnosť, redundanciu a efektivitu pri spravovaní veľkých sietí. Distribuovú architektúru riadiacej roviny možno ďalej rozdeliť na logicky centralizovanú a logicky distribuovanú.

  - **Logicky centralizovaná architektúra**
    V logicky centralizovanej architektúre si kontroléry rozdeľujú úlohy pri riadení celej siete. Každý z kontrolérov má globálny pohľad na celú sieť a prehľad o zmenách v sieti. Avšak v tejto architektúre pre jej správne fungovanie treba zaistiť adekvátnu synchronizáciu medzi kontrolérmi. Zariadenia v dátovej rovine vnímajú viacero kontrolérov ako jedinú entitu.
  - **Logicky distribuovaná architektúra**
    V logicky distribuovanej architektúre sú kontroléry fyzicky a logicky distribuované. To znamená že, každý kontrolér má na starosti riadenie iba určitý segment v sieti. Kontrolér tak poskytuje abstraktný pohľad iba na svoj segment v sieti, kde na rozdiel od logicky centralizovanej architektúry majú všetky kontroléry globálny pohľad na celú sieť.

  Z pohľadu umiestienia kontrolérov v sieti možno rozdeliť fyzicky distribuovanú architektúra na plochú (flat) a hierarchickú.

  - **Plochá architektúra**
    V plochej alebo horizontálnej architektúre sú kontroléry medzi sebou hierarchicky rovnocenné a tak riadiaca rovina pozostáva z jednej vrstvy. Kontroléry medzi sebou zdieľajú rovnaké úlohy, ale majú limitovaný pohľad na sieť v rámci svojej spravovanej domény.
  - **Hierarchická architektúra**
    V hierarchickej alebo vertikálnej architektúre panuje medzi kontrolérmi určitá hierarchia. Zvyčajne je riadiaca rovina rozdelená do dvoch alebo troch vrstiev, pričom kontroléry v rôznych vrstvách majú rozličné úlohy a rozličný prehľad o sieti. Najnižšia z hierarchických vrstiev obsahuje lokálne kontroléry, zatiaľ čo najvyššia vrstva obsahuje zvyčajne jeden hlavný (root) kontrolér.

  Využitie plochej a hierarchickej architektúry má svoje výhody a nevýhody. U oboch architektúr je možné docieliť zníženie latencie pri odosielaní inštrukcií pre zariadenia v dátovej rovine. Plochá architektúra poskytuje väčšiu

odolnosť voči výpadkom siete pri zlyhaní kontroléru, ale ovládanie a synchronizácia kontrolérov sa stáva náročnejšou. V hierarchickej architektúre je jednoduchšie riadenie kontrolérov, ale opäť v nej nastáva problém s jediným bodom zlyhania, keďže najvyššia vrstva hierarchickej riadiacej roviny obsahuje zvyčajne len jeden kontrolér.

Architektúry, ktoré používajú viacero kontrolérov, ešte možno rozlíšiť na statické a dynamické.

- **Statická architektúra**
  V statickej architektúre majú kontroléry a sieťové zariadenia pevne dané usporiadania a role, čo dáva vyššiu mieru stability v sieti.
- **Dynamická architektúra**
  V dynamickej architektúre sa môže meniť usporiadanie a väzby medzi kontrolérmi a sieťovými zariadeniami. To prináša do siete väčšiu flexibilitu.

Celkový počet kontrolérov ako aj výber ich umiestnenia v distribuovanej architektúre ovplyvňuje výkon riadiacej roviny. Preto pri výbere počtu SDN kontrolérov a návrhu optimálneho riešenia pre riadenie siete sa berú do úvahy nasledujúce parametre [14]:

- Požadovaná doba odozvy medzi kontrolérom a SDN zariadeniami, ako aj doba odozvy medzi kontrolérmi
- Požadovaná dostupnosť, šírka pásma, spoľahlivosť a pod.
- Topológia siete

3.3 Sieťové prvky v SDN

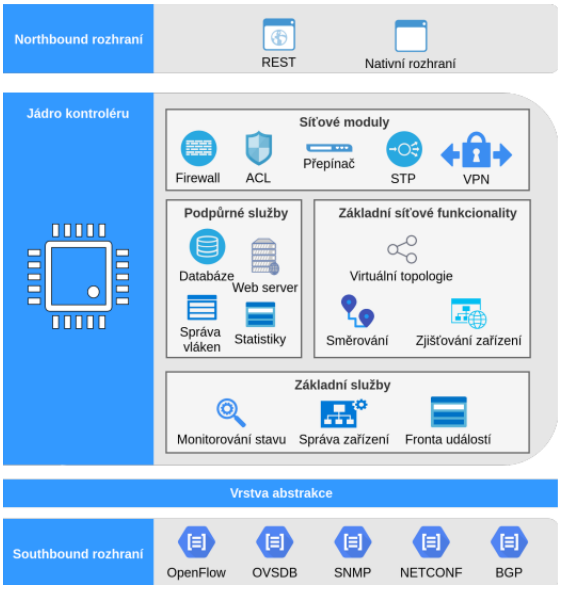
3.3.1 Prepínače

V literatúre sa často označujú zariadenia dátovej roviny ako prepínače. Pričom sa jedná o zariadenia (prepínače či smerovače), ktoré dokážu fungovať nezávisle na kontroléri. SDN prepínače môžu mať hardvérovú alebo softvérovú podobu. Softvérové prepínače sú umiestnené na serverovej platforme a vytvárajú spojenia medzi inými sieťovými prvkami a aplikáciami [7]. Hardvérové prepínače SDN dokážu vykonávať svoju pôvodnú úlohu (prepínanie, smerovanie) ako aj podporovať niektorý z protokolov južného rozhrania (napr. OpenFlow). Základná štruktúra prepínačov v SDN obsahuje flow tabuľky ktoré, ako bolo uvedené, určujú ako spracovať dátový tok.

3.3.2 SDN kontroléry

SDN kontrolér má softvérovú podobu a ako už bolo popísané, ovláda sieťové prvky v určitej doméne. Sieťové prvky tak iba prijímajú inštrukcie od SDN kontroléra. Zvyčajne býva SDN kontrolér umiestnený na fyzickom či virtuálnom serveri, ktorý poskytuje dostatočný výpočtový výkon a podporuje moderné technológie ako virtualizácia a vysoká dostupnosť. Vďaka virtualizácií možno dynamicky meniť výpočtové prostriedky pre kontrolér, a tým pádom prispôsobovať kontrolér k aktuálnemu zaťaženiu siete. Vysoká dostupnosť pridáva určitú mieru redundancie a znižuje riziko výpadku v sieti pri zlyhaní serveru, na ktorom je spustený kontrolér [8].

Najdôležitejšiou súčasťou architektúry SDN kontroléru sú sieťové moduly. Tie poskytujú základné aj pokročilé sieťové funkcionality. Ďalšou dôležitou súčasťou SDN kontroléra sú rozhrania pre komunikáciu so sieťovými prvkami a aplikáciami. Detailnejší prehľad architektúry SDN kontroléra je zobrazený na obrázku 3.2.



Obr. 3.2: Schéma SDN kontroléra [8]

Funkcionality kontroléra sú naprogramované skrz moduly alebo aplikácie.

- **Moduly** - nachádzajú sa v jadre kontroléra a sú napísané v rovnakom programovacom jazyku, aký používa kontrolér. Moduly komunikujú medzi sebou priamo a dosahujú tak najvyššej možnej rýchlosti spracovania požiadavok [8].
- **Aplikácie** - nachádzajú sa mimo kontrolér. Môžu byť napísané v ľubovoľnom programovacom jazyku nezávisle od programovacieho jazyka, ktorý využíva

kontrolér. Komunikácia s aplikáciami prebieha cez severné rozhranie. Aplikácie sa používajú pre pokročilejšie funkcie, u ktorých nevadí nižšia rýchlosť spracovania [8].

Sieťové moduly by mali vykonávať iba jeden druh funkcionality. Následne podľa nutnosti dokážu SDN kontroléry aktivovať alebo deaktivovať potrebné sieťové moduly. Medzi niektoré zo základných sieťových modulov možno zaradiť [8]:

- **ACL** - jednoduchý mechanizmus pre filtrovanie správ.
- **L3 smerovač** - zaisťuje preposielanie paketov podľa IP adries.
- **L2 prepínač** - zaisťuje preposielanie rámcov podľa MAC adries.
- **L2 protokoly na zabránenie vzniku slučiek** - obsahuje protokoly podobné STP, ktoré zabraňujú vzniku L2 slučiek v sieti.

Medzi pokročilé funkcie, ktoré vykonávajú sieťové moduly možno zaradiť [8]:

- **AAA funkcionalita** - SDN kontroléry používajú pri komunikácií s užívateľom bezpečnostné princípy autentizácie, autorizácie a účtovania.
- **GUI** - kontroléry umožňujú aspoň základnú správu siete prostretníctvom grafického rozhrania, ktoré je najčastejšie implementované vo webovom formáte, a tak je prístupné rôznym druhom koncových zariadení.
- **Zber štatistík** - kontrolér umožňuje sledovať stav topológie siete a vytvárať rôzne štatistiky o dátových tokoch.
- **Zisťovanie topológie** - kontrolér dokáže vytvoriť prehľad o pripojených zariadeniach v dátovej rovine. K tomu kotrolér využíva napr. protokol LLDP.

SDN kontroléry sa ďalej delia na dva základné druhy: open-source a komerčné kontroléry [8].

- **Open-source** - jedná sa o voľne dostupné kontroléry, ktoré možno používať zdarma. V súčasnosti existuje veľké množstvo voľne dostupných kontrolérov. Tie zahrňujú experimentálne kontroléry ako aj kontroléry, ktoré sú vhodné pre nasadenie do sietí spoločností. Open-source kontroléry sú vyvíjané komunitou a zdrojový kód kontroléru je voľne prístupný verejnosti. Tieto kontroléry používajú otvorené rozhrania, a preto sú kompatibilné so širokou škálou sieťových zariadení.

- **Komerčné** - tieto kontroléry vyvíjajú korporácie a sú dostupné za určitý poplatok. Kontroléry bývajú často založené na open-source kontroléroch, ktorým výrobcovia pridávajú alebo modifikujú ich funkcionalitu. Súčasťou kúpi komerčného kontroléra býva aj jeho profesionálna podpora zo strany spoločnosti. Komerčné kontroléry veľmi často používajú proprietárne protokoly, ktoré umožňujú komunikáciu s úzkou škálou sieťových zariadení. Jedná sa hlavne o zariadenia, ktoré vyrába ten istý výrobca.

3.3.3 Aplikácie

Aplikácie implementujú dodatočnú riadiacu logiku, ktorú kontrolér prekladá do príkazov pre zariadenia v dátovej rovine. SDN možno nasadiť v akomkoľvek tradičnom sieťovom prostredí, od podnikových sietí až po dátové centrá. Takáto rôznorodosť prostredí viedla k vytvoreniu širokej škáli sieťových aplikácií. Existujúce sieťové aplikácie vykonávajú tradičné funkcie (load-balancing, presadzovanie QoS) alebo pokročilejšie funkcionality ako virtualizácia siete či riadenie mobility v bezdrôtových sieťach. Očakáva sa, že rozmanitosť sieťových aplikácií bude jedným z pilierov prijatia SDN ako sieťovej architektúry v podnikových sieťach či datacentrách [8].

3.4 OpenFlow

Protokol OpenFlow patrí v súčasnosti medzi najpoužívanejší protokol južného rozhrania. Tvorí dôležitú súčasť SDN, keďže umožňuje SDN kontroléru komunikovať s fyzickými alebo virtuálnymi zariadeniami v dátovej rovine. Od svojho vzniku v roku 2009, kedy bola vydaná jeho prvá verzia 1.0 až po aktuálnu verziu 1.5.1 prešiel tento protokol mnohými modifikáciami a vylepšeniami [15]. Problematika protokolu OpenFlow je rozsiahla, preto je v rámci dimplomovej práce uvedený stručný prehľad protokolu. Obrázok 3.3 zobrazuje štruktúru prostredia OpenFlow, ktorá je popísaná nižšie.

3.4.1 Komunikácia a rozhrania v OpenFlow

Správy protokolu Openflow využívajú protokol TCP alebo jeho zabezpečenú verziu TLS. Z bezpečnostných dôvodov sa odporúča používať minimálne verziu TLS 1.2. Prepínač zvyčajne komunikuje s jedným, prípadne viacerými SDN kontrolérmi. Správy v protokole OpenFlow sa delia na tri typy: kontrolér-prepínač, symetrické a asynchrónne [16].

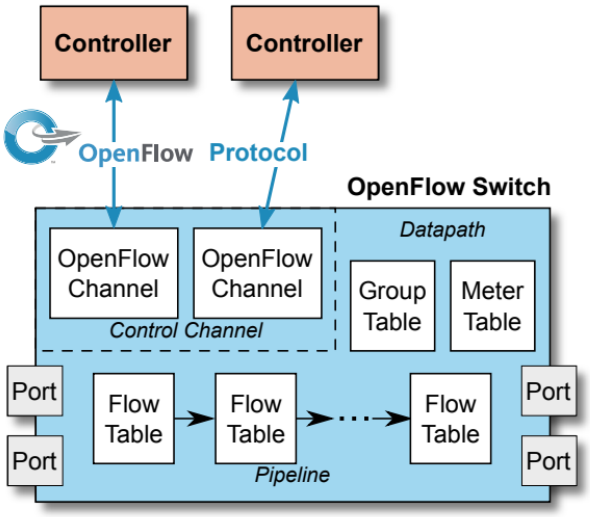
- **Kontrolér-prepínač** - typ správ, kde komunikáciu zahajuje kontrolér. V závislosti na druhu správy prepínač musí alebo nemusí odpovedať kontroléru.

Zvyčajne sa jedná o správy, ktoré ovládajú prepínač alebo zisťujú jeho stav.

- **Symetrické** - správy, ktoré odosiela prepínač alebo kontrolér bez toho, aby reagoval na predošlú žiadosť. Tento druh správ sa používa pri vytváraní a overovaní spojenia.
- **Asynchrónne** - tieto správy sa posielajú bez toho, aby ich kontrolér vyžiadal od prepínača. Prepínače posielajú asynchrónne správy kontroléru napr. pri preposlaní paketu alebo pri zmene stavu prepínača.

Rozhrania, na ktorých prepínač prijíma pakety definuje OpenFlow ako vstupné (ingress) rozhrania. Analogicky rozhrania, z ktorých sa odosielajú pakety označuje OpenFlow ako výstupné (outgoing). Ďalej OpenFlow delí rozhrania prepínača na fyzické, logické a rezervované [16].

- **Fyzické rozhrania** - referujú priamo k rozhraniam prepínača.
- **Logické rozhrania** - nereferujú priamo ku konkrétnemu rozhraniu prepínača, ale k abstrakcií rozhraní ako napr. tunelové alebo loopback rozhrania.
- **Rezervované rozhrania** - úlohou rezervovaných rozhraní je vykonávať špecifické úlohy, ako napr. preposielanie správ SDN kontroléru, flooding alebo klasické prepínanie správ bez nutnosti použitia OpenFlow protokolu. Rezervované rozhrania sa ďalej delia na požadované a voliteľné, pričom požadované rozhrania musí podporovať každý OpenFlow prepínač.



Obr. 3.3: Schéma Openflow prepínača [16]

3.4.2 Tabuľky v OpenFlow

Štruktúra OpenFlow ďalej pozostáva z troch typov tabuliek: tabuľky tokov, tabuľky skupiny a tabuľky meraní [16].

- **Tabuľka meraní (meter table)** - obsahuje záznamy, vďaka ktorým možno monitorovať jednotlivé dátové toky. Tabuľka meraní môže slúžiť napr. pre dohľad na QoS.
- **Tabuľka skupiny (group table)** - určité dátové toky možno zjednotiť a spravovať jedným záznamom z tabuľky skupiny. Tabuľky skupiny obsahujú jednu sadu inštrukcií pre správu zjednotených dátových tokov.
- **Tabuľka toku (flow table)** - obsahuje záznamy o dátových tokoch. Tie pozostávajú z informácií, ako identifikovať dátový tok a inštrukcií, ako ho následne spracovať.

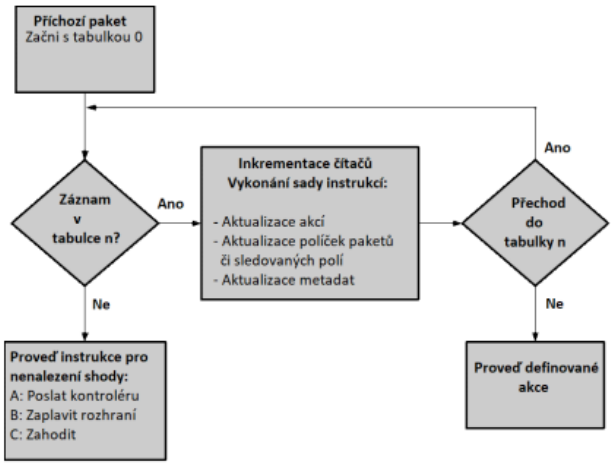
Prepínače obsahujú jednu alebo viac tabuliek toku, ktoré sú zreťazené (tvoria pipeline). Každá z flow tabuliek má unikátne číselné označenie a jednotlivé tabuľky sú zoradené vzostupne začínajúc od nuly. Tabuľky toku obsahujú polia, vďaka ktorým je možné detekovať určitý dátový tok a určiť, ako ho spracovať. Samotný pojem dátového toku možno definovať ako postupnosť paketov s podobnými vlastnosťami. Takéto pakety zdieľajú rovnaké zdrojové a cieľové IP adresy, VLAN tagy a pod. Prvá verzia protokolu OpenFlow 1.0 využívala tri polia v tabuľke tokov. S novšími verziami pribúdali dodatočné polia a v poslednej verzii 1.5.1. tak tabuľka tokov obsahuje celkovo sedem polí [16]:

- **Pravidlá (match fields)** - obsahuje sadu informácií, podľa ktorých sa hľadá zhoda s paketmi. Tieto údaje zahrňujú napr. vstupné a výstupné rozhrania na SDN prepínači, zdrojové a cieľové IP a MAC adresy, porty protokolov transportnej vrstvy atď.
- **Priorita (priority)** - určuje prioritu každej flow tabuľky. Každá flow tabuľka má svoje unikátne priority číslo.
- **Počítadlá (Counters)** - používajú sa na sledovanie štatistík daného toku a aktualizovaní hodnôt pri spracovaní paketu.
- **Inštrukcie (instructions)** - definuje sadú akcií, ktoré treba vykonať pri nájdení zhody s dátovým tokom. Zvyčajne sa jedná o presunutie paketu na výstupné rozhranie, preposlanie paketu kontroléru, presun paketu k inej tabuľke tokov alebo zahodenie paketu.

- **Časovače (timeouts)** - tabuľka tokov zvyčajne obsahuje dva druhy časovačov (hard a idle timeout). Ak niektorý z časovačov pretečie, vymaže sa príslušná tabuľka tokov.
- **Cookie** - pole využívané SDN kontrolérom pre filtrovanie štatistík dátového toku, prípadne na úpravu alebo zmazanie tabuľky tokov.
- **Flags** - pole určuje, ako spravovať jednotlivé položky v dátovom toku.

3.4.3 Spracovanie paketov v OpenFlow

Potom, ako OpenFlow prepínač príjme paket na vstupnom rozhraní dochádza k porovnávaniu paketu so všetkými tabuľkami tokov. Ak sa nájde zhoda medzi paketom a tabuľkou toku, prepínač následne vykoná inštrukcie pre spracovanie paketu podľa príslušnej tabuľky toku a aktualizuje údaje v poli counters. Pri nenájdení žiadnej zhody medzi tabuľkami toku (table-miss) zvyčajne nastávajú nasledujúce situácie: prepínač odošle paket kontroléru alebo sa paket zahodí [19]. Zjednodušená podoba tohto algoritmu je zobrazená na obrázku 3.4.



Obr. 3.4: Vývojový diagram spracovania paketov v OpenFlow [19]

OpenFlow protokol definuje dva prístupy ako spravovať tabuľky tokov v prepínačoch. Jedná sa o proaktívny a reaktívny režim, pričom oba režimy je možné medzi sebou kombinovať [20].

- **Proaktívny režim** - v tomto režime uchováva prepínač trvalo tabuľky toku v pamäti. Tabuľky tak ostávajú v pamäti ešte predtým, ako ich bude nutné použiť. Proaktívny režim je pamäťovo náročnejší, keďže potrebuje uchovať väčšie množstvo tabuliek toku, ale menej zaťažuje kontrolér.

- **Reaktívny režim** - v tomto režime sa tabuľky tokov pridávajú do prepínača až v prípade nutnosti, kedy nastane table-miss. Staršie alebo neopoužívané tabuľky toku sa za použitia časovačov priebežne vymazávajú. Režim menej zaťažuje pamäť, ale je viac závislý na kontroléri. Ak je kontrolér zaťažený, nová tabuľka toku je vytvorená s oneskorením.

3.5 Zhrnutie vlastností SDN

SDN vďaka oddeleniu dátovej a riadiacej roviny a centralizovaní riadenia odstraňuje väčšinu problémov tradičných počítačových sietí. Ako už bolo zmienené, v tradičných sieťach je priame spojenie medzi riadiacou a dátovou rovinou. To sťažuje vývoj a zavádzanie nových sieťových funkcií (napr. smerovacích algoritmov), pretože by to znamenalo úpravu riadiacej roviny všetkých sieťových zariadení. Či už prostredníctvom inštalácie nového firmvéru alebo v niektorých prípadoch aj modernizáciou hardvéru. V SDN je vďaka softvérovému riadeniu celej siete integrácia a vývoj nových funkcionalít siete jednoduchšia a funkcie middleboxov nahrádzajú aplikácie. Medzi výhody SDN možno zaradiť [17], [18]:

- **Centralizované riadenie** - centralizovanie riadenia siete do jednej entity uľahčuje správu a konfiguráciu siete. Sieťovým administrátorom tak umožňuje jednoduchšie definovať a presadzovať sieťovú politiku, čoho výsledkom je vyššia sieťová bezpečnosť, výkon a spoľahlivosť.
- **Programovateľnosť** - v SDN sú sieťové zariadenia v dátovej vrstve programovateľné a možno počas ich prevádzky meniť ich konfiguráciu, aby vyhovovala dynamickým požiadavkám siete. To umožňuje správcom siete rýchlo prispôsobiť sieť na meniace sa požiadavky siete, čo vedie k zvýšeniu výkonu a efektívnosti siete.
- **Vylepšená bezpečnosť** - centralizované riadenie sieti v SDN umožňuje nastavovať bezpečnostnú politiku pre celú sieť z jedného miesta, čo umožňuje jednoduchšie detekovať a reagovať na bezpečnostné hrozby.
- **Škálovateľnosť** - SDN uľahčuje škálovanie siete, aby vyhovovala meniacim sa požiadavkám na prevádzku siete. Vďaka možnosti programového riadenia siete môžu správcovia rýchlo upraviť správanie siete tak, aby bez potreby manuálneho zásahu sa sieť dokázala prispôsobiť aktuálnemu zaťaženiu.
- **Nezávislosť na výrobcovi** - pre komunikáciu so širokou škálou sieťových zariadení a kontrolérom stačí jeden protokol. To prináša voľnosť pri výbere sieťových zariadení a eliminuje tak závislosť zaobstarávať sieťové zariadenia od jedného výrobcu.

- **Jednoduchšia správa siete** - centralizované riadenie siete v SDN poskytuje správcom siete abstraktný pohľad na sieťové zariadenia umiestnené v dátovej rovine. Táto abstrakcia vedie k jednoduchšej správe siete a rýchlešiemu odstraňovaniu chýb v konfigurácií (troubleshooting), čo vedie k lepšej dostupnosti a spoľahlivosti siete.

Vďaka SDN je možné vytvárať ľahko škálovateľnú a programovateľnú sieť, čo umožňuje spoločnostiam jednoduchšie prispôsobovať sieť na aktuálnu záťaž. SDN však so sebou prináša aj určité obmedzenia [17]:

- **Náročnosť** - SDN môžu byť zložitejšie ako tradičné siete, pretože zahŕňajú komplexnejší súbor technológií, znalostí a zručností pre správu siete. Používanie centralizovaného riadenia na správu siete vyžaduje hlboké pochopenie architektúry a protokolov SDN.
- **Závislosť na kontroléri** - centralizovaný kontrolér je kritickou súčasťou siete a pri jeho zlyhaní dôjde k ochromeniu siete. Je preto potrebné vykonať protiopatrenia, aby SDN kontrolér nepredstavoval jediný bod zlyhania v sieti.
- **Kompatibilita** - niektoré staršie sieťové zariadenia nemusia byť kompatibilné s SDN, čo znamená, že spoločnosti budú musieť tieto zariadenia vymeniť alebo aktualizovať, aby mohli nasadiť SDN.
- **Bezpečnosť** - SDN môže navýšiť bezpečnosť siete, ale taktiež táto technológia prináša nové bezpečnostné riziká. Napríklad kontrolér ako jediný bod riadenia siete je atraktívny cieľ pre útočníka. Programovateľnosť siete tiež umožňuje útočníkom jednoduchšie manipulovať správanie siete.
- **Latencia** - zhotovenie a preposlanie inštrukcií od kontroléru pre SDN zariadenia má určité oneskorenie. Avšak s väčším počtom SDN zariadení bude toto oneskorenie narastať, čo ovplyvňuje výkonnosť siete.

Medzi SDN a tradičnými sieťami sú aj bezpečnostné rozdiely. Vďaka pohľadu na celú sieť a možnosti definovať bezpečnostné zásady pre celú sieť ponúka SDN v mnohých ohľadoch lepšie zabezpečenie. Keďže však softvérovo definované siete využívajú centralizovaný kontrolér, zabezpečenie kontroléra je kľúčové pre udržanie bezpečnosti siete. Opis bezpečnostných hrozieb v SDN je uvedený v nasledujúcej subkapitole.

3.6 Bezpečnosť SDN

S oddelením jednotlivých rovín v architektúre SDN nastávajú nové bezpečnostné riziká, ktoré cielia na logické roviny a komunikačné rozhrania SDN. Nižšie sú uve-

dené niektoré hrozby, ktoré narušujú bezpečnostné princípy integrity, dostupnosti a dôvernosti v SDN.

DoS/DDoS útok

Cieľom DoS alebo DDoS útokov je prostredníctvom nelegitímnych správ zahltiť SDN kontrolér či sieťové zariadenia a tým obmedziť dostupnosť siete a služieb. Existuje viacero typov DoS/DDoS útokov, ktoré mieria na SDN prostredie.

V prípade DoS/DDoS útoku na kontrolér, je kontrolér zahltený falošnými flow žiadosťami od útočiaceho zariadenia. Výpočtové zdroje kontroléru spracovávajú falošné žiadosti, zatiaľ čo legitímne flow žiadosti sú ignorované. Flow záznamy v sieťových prvkoch majú určitú časovú platnosť, takže po krátku dobu dokážu sieťové prvky fungovať bez interakcie s kontrolérom. Avšak po skončení tejto platnosti nastáva kolaps siete, keďže sieťové zariadenia nedostávajú aktualizácie, ako smerovať dátové toky [21].

DoS/DDoS útoky tiež cielia na sieťovú infraštruktúru v dátovej rovine. Jednou možnosťou tohto útoku je vytvárať veľké množstvo falošných flow záznamov, ktoré zahltia pamäť zariadenia. Zariadenie tak nedokáže uložiť legitímne záznamy v pamäti a kontrolér tak stráca možnosť riadiť tento sieťový prvok [21].

S OpenFlow, prípadne iným štandardom južného rozhrania, musia SDN prepínače uchovávať pakety, pre ktoré nemajú flow záznam vo vyrovnávacej pamäti. Veľkým množstvom falošných paketov možno zaplniť vyrovnávaciu pamäť zariadenia čo spôsobí, že zariadenie začne zahadzovať legitímne pakety, pre ktoré je potrebné vytvoriť flow záznam [21].

Link Fabrication útok

Cieľom tohto útoku je pretvárať spojenia a topológiu siete, ako jú vidí kontrolér. Útočník môže napr. vytvoriť fingované priame spojenie medzi dvoma sieťovými zariadeniami, ktoré bude kontrolér vnímať ako legitímne. To umožňuje vložiť do siete infikovaný uzol a sprostredkovať man-in-the-middle útok. Útoky najmä zneužívajú činnosti discovery protokolov ako je napr. protokol LLDP. Bez zaistenia autentizácie LLDP paketov útočník prostredníctvom falošných LLDP paketov dokáže docieliť preusporiadanie topológie siete z pohľadu kontroléru [23].

Hijacked/Rogue kontrolér

SDN kontrolér predstavuje najzraniteľnejší bod v architektúre SDN, keďže ako jediný uzol spravuje riadenie sieťových zariadení. Ak sa útočníkovi podarí získať prístup do kontroléru, dokáže tak ovládať všetky prebiehajúce procesy v sieti. Útočník môže cielene meniť flow záznamy, aby napr. zabránil smerovaniu paketov

k určitému uzlu. V ďalšom prípade môže útočník nasmerovať pakety na jeden cielený uzol. Prostredníctvom neho môže útočník sprostredkovať man-in-the-middle útok alebo vytvoriť tzv. blackhole uzol. Útočníkovi sa tak naskytne príležitosť zahadzovať, meniť alebo zisťovať obsah prijatých paketov na zariadení [21] [22].

Ďalším možným útokom na riadiacu rovinu SDN je nasadenie škodlivého SDN kontroléra do siete. So škodlivým kontrolérom dokáže útočník ovplyvniť či prípadne pozastaviť činnosti legitímnych kontrolérov a manipulovať správy z aplikačnej roviny [21].

Akýkoľvek útok smerujúci na kontrolér v SDN architektúre môže mať devastujúce účinky. Zatiaľ čo centralizované riadenie je z hľadiska správy siete výhodné, v prípade útoku dokáže ľahko ohroziť integritu riadiacich alebo aplikačných správ, dostupnosť sieťových služieb a dôvernosť dát z aplikačnej roviny.

Škodlivé aplikácie

Vzhľadom k tomu, že aplikácie v SDN sú často vytvárané prostredníctvom tretej strany, predstavuje tento prístup potenciálnu hrozbu. Škodlivé aplikácie môžu mať na sieť podobné následky ako škodlivý kontrolér. Napr. napadnutá aplikácia, ktorej úlohou je vykonávať detailnú inšpekciu paketov, predstavuje hrozbu pre sieť. Zistené dáta z inšpekcie paketov nepriamo ohrozujú riadenie siete, keďže poskytujú útočníkovi dodatočné informácie, kam cieliť ďalší útok [21].

Veľký objem dát s informáciami o sieti a fakt, že sú umiestnené v SDN kontroléri, jedinom centrálnom riadiacom bode, poskytuje škodlivým aplikáciam možnosť narušenia ich integrity a dôvernosti. Preto je potrebné zaistiť princípy autentizácie a autorizácie pri komunikácií s SDN kontrolérom. Každá aplikácia by tak mala mať stanovené, ku akým dátam v SDN kontroléri má oprávnený prístup [21].

Control-Data Plane Link útoky

Ďalšou kľúčovou oblasťou v SDN architektúre, ktorá predstavuje zraniteľný bod, je spojenie medzi riadiacou a dátovou rovinou. OpenFlow, ako najpoužívanejší protokol južného rozhrania, má voliteľné použitie protokolu TLS pre zaistenie bezpečnosti dát. Ale pri jeho nepoužití je južné rozhranie náchylné na man-in-the-middle a tzv. blackhole útok [21] [22].

V man-in-the-middle útoku útočník nasadzuje škodlivý uzol umiestnený medzi SDN kontrolérom a zariadeniami v dátovej rovine. Prostredníctvom tohto uzlu môže útočník meniť obsah riadiacich informácií alebo sledovať ich obsah [21].

V prípade blackhole útoku útočiaci uzol cielene zahadzuje všetky prijaté inštrukcie. To spôsobí stratu spojenia medzi SDN kontrolérom a dátovou rovinou.

Ak sa útočníkovi podarí nasadiť medzi dátovú a riadiacu rovinu infikovaný uzol, dokáže tak ohroziť činnosť celej siete. Man-in-the-middle útok cieli priamo

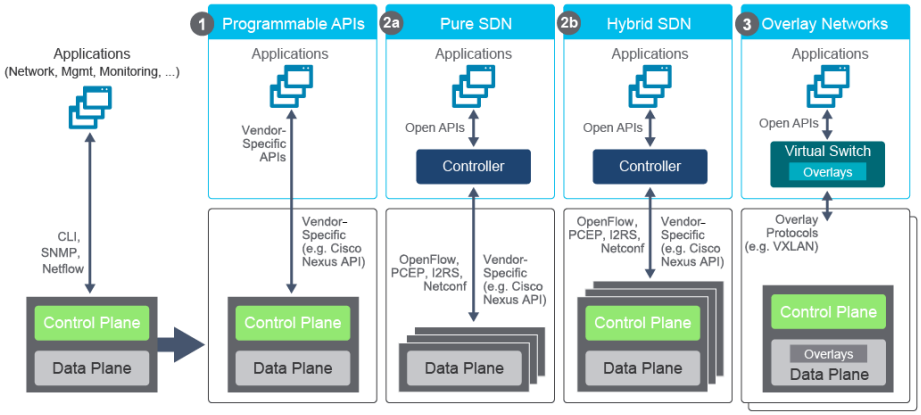
na integritu riadiacich dát. Útočník úpravou riadiacich správ dokáže ovplyvniť prevádzku siete vo svoj prospech. Útok typu blackhole cieli na dostupnosť sieťových služieb. Ak riadiace informácie nie sú predávané z kontroléru do sieťových zariadení, hrozí tak celkový výpadok siete [21].

Eavesdropping útoky

Útočník, ktorý sa snaží napadnúť alebo získať neoprávnený prístup do SDN siete môže najprv vykonať odpočúvanie (neoprávnené sledovanie a inšpekcia paketov) na niektorom spojení v sieti. Takýto útok umožňuje zisťovať citlivé informácie o sieti, ktoré možno využiť v prospech ďalšieho útoku. V SDN možno odpočúvanie vykonať medzi zariadeniami dátovej roviny alebo medzi spojením medzi dátovou a riadiacou rovinou. Odpočúvanie priamo neohrozuje dostupnosť, dôvernosť a integritu dát, ale poskytuje informácie, ako ďalej narušiť tieto aspekty bezpečnosti [21].

3.7 Možnosti nasadenia programovateľnosti do sietí

V súčasnosti existuje viacero techník a spôsobov, ako zaviesť programovateľnosť do sietí. Niektoré z možností sú ilustrované na obrázku 3.5, kde ľavá časť obrázku predstavuje tradičnú sieť spravovanú prostredníctvom CLI alebo SNMP.



Obr. 3.5: Možnosti nasadenia programovateľnosti do siete [1]

V počiatkoch vývoja SDN výrobcovia sieťových zariadení predovšetkým vytvárali pre svoje zariadenia proprietárne API. Pomocou nich a externých aplikácií bolo možné programovať sieťové zariadenie. Podobne ako v tradičných sieťach obsahovali sieťové zariadenia dátovú a riadiacu rovinu [1].

Pure SDN predstavuje koncept, ktorý bol doteraz uvažovaný v diplomovej práci. V pure SDN je riadenie centralizované v kontroléri a sieťové zariadenia neobsahujú riadiacu rovinu. Ako už bolo uvedené, tento prístup ma oproti tradičným sieťam množstvo výhod. Plný prechod z tradičných sietí na SDN je ale komplikovaný a vyžaduje značné množstvo financií a komplexnú znalosť problematiky SDN. Preto je v praxi najviac uplatňovaný hybridný model SDN [1].

Hybridné SDN prestavujú sieť, v ktorej koexistujú tradičná aj SDN sieťová architektúra. Sieť je spravovaná prostredníctvom distribuovanej riadiacej roviny ako aj SDN kontrolérom. Obe riadiace roviny tak spravujú určité funkcionality v sieti [1].

Ďalšou možnosťou ako docieliť programovateľnosť v sieti je použitie tzv. overlay sietí. Overlay sieť je virtuálna sieť vytvorená nad fyzickou sieťou, ktorá je programovateľne spravovaná pomocou softvéru [1]. Samotná komunikácia medzi dvoma uzlami vo virtuálnej sieti zvyčajne používa niektorý z tunelovacích protokolov (VxLAN, GRE, GENEVE...).

3.8 Použitie SDN

Táto krátka subkapitola popisuje niektoré zo sieťových riešení, ktoré tvoria podmnožinu ku SDN, respektíve ktoré využívajú princípy SDN.

SD-WAN

Jednou z úloh tradičných WAN sietí je pripájať používateľov na pobočke alebo v kampuse k aplikáciám umiestnených na serveroch v dátovom centre. Typicky sa na zaručenie bezpečnosti a spoľahlivého pripojenia používa protokol MPLS. S príchodom cloudu však tento spôsob pripájania nemusí byť dostačujúci [24].

SD-WAN (Software-defined Wide Area Network) je technológia, ktorá umožňuje správu WAN sietí prostredníctvom softvéru, a nie prostredníctvom hardvérových sieťových zariadení. SD-WAN umožňuje väčšiu flexibilitu, úsporu finančných nákladov či lepší výkon siete. SD-WAN využíva rôzne techniky, ako napr. šifrovanie či zavedenie kvality služieb (QoS) na optimalizáciu a zabezpečenie WAN pripojení. Softvérovo orientovaný prístup robí WAN agilnejšiu a pohotovejšiu v porovnaní s tradičnou WAN. Ako podniky presúvajú viac aplikácií a infraštruktúry do cloudu, SD-WAN sa stáva kľúčovou technológiou zabezpečujúcu konektivitu [24].

Jedným z dôvodov potreby posilnenia infraštruktúry WAN je napr. prechod na nové možnosti poskytovania služieb, ako sú SaaS (Software as a Service), IaaS (Infrastructure as a Service) a PaaS (Platform as a Service) [25].

SD Access

SD Access je riešenie, ktoré prináša koncepty softvérovo definovaných sietí a virtualizácie do prístupovej (access) vrstvy siete (časť siete, kam sa pripájajú koncoví užívatelia). SD-Access nie je nová technológia, skôr sa ale jedná o sadu nástrojov, ktoré zjednodušujú správu prístupovej vrstvy siete, koncových zariadení a užívateľov. SD-Access navyše zvyšuje bezpečnosť siete napr. jej segmentovaním do rôznych zón alebo vytváraním virtuálnych sietí [26].

SD Datacenter

Softvérovo definované dátové centrum (SDDC) alebo virtuálne dátové centrum je termín používaný na označenie dátového centra, v ktorom je celá infraštruktúra virtualizovaná a poskytovaná ako služba, čo je známe pod pojmom ako Infrastructure as a service (IaaS). Kontrola a správa softvérovo definovaného dátového centra je plne automatizovaná a udržiavaná prostredníctvom softvéru [27].

Kapitola 4

Virtualizácia a virtualizačné nástroje

Táto kapitola v krátkosti definuje pojem virtualizácie a popisuje virtualizačné nástroje používané v Algotechu. Konkrétne sa jedná o nástroje vSphere, ktorého dôležitou súčasťou sú ESXi hypervisor a vCenter server, a NSX-T. Problematika týchto virtualizačných platforiem je rozsiahla, preto v diplomovej práci je uvedený jemný úvod do týchto platforiem. Subkapitola o vSphere popisuje architektúru a súčasti tohto nástroja. V subkapitole o NSX-T je uvedená architektúra a komponenty NSX-T a v krátkosti rozobraná problematika prepínania, smerovania a bezpečnosti v NSX-T.

Virtualizácia je kľúčová technológia v dátových centrách, ktorá umožňuje abstrahovať zdroje fyzických zariadení, a tým ich efektívne využívať. Dátové centrá zvyčajne pozostávajú z troch hlavných virtualizačných komponentov: serverovej, sieťovej a úložiskovej virtualizácie [28].

- Serverová virtualizácia je už známy pojem, kde pomocou typ 1 hypervisora je možné abstrahovať výpočtové zdroje fyzického serveru, a tým vytvárať virtuálne servery s rôznymi operačnými systémami a výpočtovými zdrojmi. Vďaka serverovej virtualizácií je možné efektívne využiť výpočtové zdroje fyzického serveru. V súčastnosti existuje množstvo nástrojov pre serverovú virtualizáciu ako napr. Microsoft Hyper-V, Citrix XenServer alebo KVM.

- Virtualizácia úložiska (storage virtualization) analogicky k serverovej virtualizácií abstrahuje fyzické zdroje úložiska, čím je možné ho logicky zoskupovať a spravovať centrálne. S virtualizovaným úložiskom možno taktiež podľa potreby dynamicky alokovať potrebnú veľkosť úložiska pre aplikácie. Príkladom virtualizačnej platformy na virtualizáciu úložiska je vSAN od spoločnosti VMware.

- Sieťová virtualizácia slúži na vytváranie virtuálnych sietí, ktoré koexistujú s fyzickou sieťou. Pomocou virtuálnych sietí je možné lepšie optimalizovať výkon siete a zlepšiť jej bezpečnosť. Medzi dostupné sieťové virtualizačné nástroje patrí napr. NSX od spoločnosti VMware.

Ako už bolo zmienené, existuje množstvo dostupných virtualizačných platforiem dostupných na trhu. Avšak medzi najpopulárnejšie virtualizačné nástroje patrí portfólio produktov od spoločnosti VMware, ktoré využíva aj spoločnosť Algotech. Praktická časť diplomovej práce sa zaoberá prácou s prostrediami vSphere a NSX-T, preto je ku týmto nástrojom uvedený ich stručný popis.

4.1 VMware vSphere

VMware vSphere predstavuje balík aplikácií a funkcií najmä pre serverovú a úložiskovú virtualizáciu, kde jeho najvýznamnejšími časťami sú VMware ESXi (typ 1 alebo bare-metal hypervisor) a VMware vCenter server. Medzi niektoré z dodatočných aplikácií a funkcionalít vSphere patria [29]:

- VMware vSphere vMotion - funkcionalita, ktorá umožňuje migráciu virtuálnych zariadení medzi servermi priamo počas ich prevádzky. Tým nedochádza k výpadkom služieb a eliminuje sa tak potreba plánovať odstávky aplikácií kvôli údržbe serveru.

- VMware vSphere Storage vMotion - je analogická funkcionalita ku vSphere vMotion, ktorá slúži na migrovanie úložiska virtuálneho zariadenia.

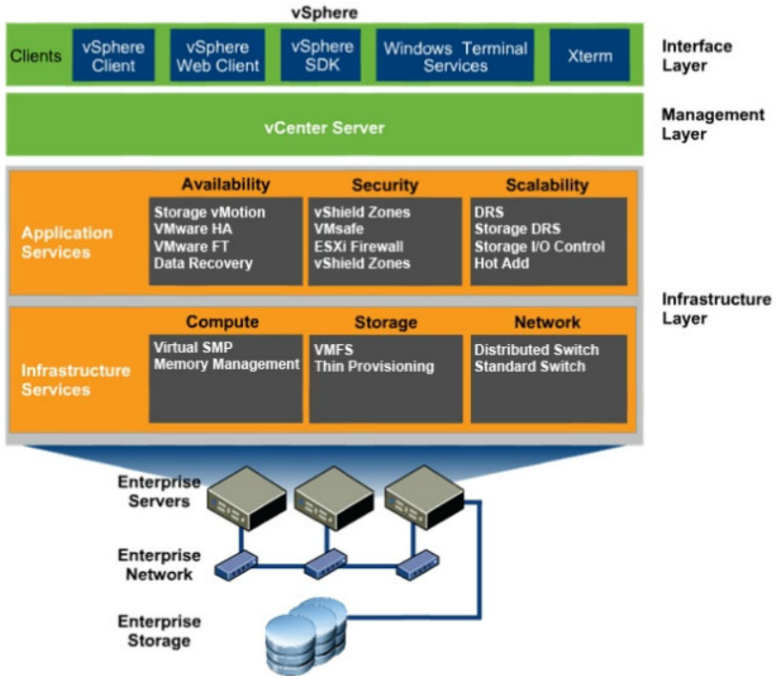
- vSphere High Availability - úlohou tejto aplikácie je zaistiť reštart a prevádzku virtuálnych zariadení na novom fyzickom serveri v prípade, ak dôjde k poruche na pôvodnom fyzickom serveri.

- vSphere Distributed Resource Scheduler (DRS) - je funkcionalita, ktorá automatizuje load-balancing virtuálnych zariadení medzi ESXi hostami v clusteri. To zaisťuje rovnomerné využívanie výpočtového výkonu medzi ESXi hostami v rámci clusteru. ESXi hosta možno definovať ako hardvérové zariadenie, na ktorom je nainštalovaný ESXi hypervisor.

- VMware vSphere Thin Provisioning - je funkcionalita, ktorá dynamicky alokuje potrebnú veľkosť úložiska pre virtuálne zariadenie.

- vSphere Distributed Switch - jedná sa o virtuálny prepínač, ktorý zabezpečuje konektivitu v rámci virtuálneho prostredia ESXi a fyzickou sieťou. Konfiguruje sa centrálne vo vCenter a následne sa jeho konfigurácia distribuuje medzi ESXi zariadenia.

Architektúra prostredia vSphere je zobrazená na obrázku 4.1. Tá pozostáva z troch vrstiev a to: vrstvy infraštruktúry, spravovania a prístupového rozhrania. Vrstva infraštruktúry pozostáva zo služieb, ktorých úlohou je abstrahovať fyzické zdroje zariadení. Medzi ne patrí aj ESXi. Súčasťou vrstvy infraštruktúry sú aj aplikácie zaručujúce škálovateľnosť, bezpečnosť a dostupnosť služieb. Spravovaciu vrstvu vSphere tvorí vCenter server. Tvorí tak centrálny bod, cez ktorý možno konfigurovať a spravovať virtuálne prostredie datacentra. Vrstva prístupového rozhrania definuje možnosti prístupu do vSphere. Užívatelia môžu pristupovať do vSphere prostredníctvom GUI ako napr. vSphere Client.



Obr. 4.1: Architektúra vSphere prostredia [30]

VMware vCenter server je nástroj pre centralizované riadenie aplikácií vo vSphere. Prostredníctvom vCenter je možné spravovať ESXi zariadenia a vytvárať clustre z ESXi hostov. Takýto cluster sa tiež nazýva vSphere cluster, čo je logické zoskúpenie viacerých ESXi hostov, ktoré môžu navzájom komunikovať a spolu agregovať výpočtové zdroje. V rámci vCenter je docielená skutočná abstrakcia od hardvéru, keďže eliminuje závislosť na umiestnení virtuálneho zariadenia na pevne danom ESXi zariadení. V prípade výpadku fyzického serveru dokáže vCenter prostredníctvom nástrojov ako napr. High Availability presunúť virtuálne zariadenie na iný ESXi server. VMWare vCenter je takisto potrebný na ovládanie aplikačných služieb vo vSphere. Vytváranie a spravovanie virtuálnych zariadení na ESXi zariadeniach alebo v ESXi clusteri je taktiež centralizované pomocou vCenter [31].

4.2 VMware NSX-T

VMware NSX je sieťová virtualizačná platforma, ktorá je súčasťou architektúry SDDC spoločnosti VMware. NSX sa dodáva v rôznych formách ako napr. [33]:

- NSX Data Center
- NSX Cloud
- NSX SD-WAN
- NSX Hybrid Connect

NSX Data Center sa následné delí na dve ďalšie varianty: NSX-V, ktorá je určený najmä na prácu s prostredím VMware VSphere a NSX-T, ktorá ponúka funkcie virtualizácie siete a kybernetickej bezpečnosti nezávisle na prostredí a softvéri, v ktorom je NSX-T nasadené (multi-hypervisor, multi-cloud, alebo multi-container). Obe varianty slúžia na virtualizáciu rôznych L2 až L7 funkcionalít, ako napr. virtuálne prepínanie a smerovanie, DHCP a DNS služby, load balancing či firewall a mikrosegmentácia [33].

NSX-T podporuje viacero druhov hypervisorov vrátane VMware ESXi, Microsoft Hyper-V a KVM. Firmy tak môžu nasadiť virtuálne počítače a aplikácie na hypervisor podľa vlastnej preferencie, pričom stále môžu využívať sieťové a bezpečnostné funkcie, ktoré poskytuje NSX-T. Podobne možno NSX-T nasadiť v multi-cloud prostrediach vrátane verejných cloudov, ako sú Amazon Web Services (AWS) a Microsoft Azure, ako aj v súkromných cloudoch [32].

NSX-T tiež podporuje aj nasadenie kontajnerových aplikácií pomocou Kubernetes, čo umožňuje použitie NSX-T na zabezpečenie siete a bezpečnosti pre kontajnerové aplikácie. Pomocou NSX-T je možné poskytnúť virtuálnu sieťovú infraštruktúru, ktorá je plne kompatibilná s prostredím Kubernetes a umožňuje implementovať rovnakú úroveň zabezpečenia a segmentácie siete ako pri virtuálnych strojoch [32].

Vďaka funkcionalitám ako mikrosegmentácia, firewall alebo VPN poskytuje NSX-T vysokú úroveň zabezpečenia siete. Spoločnosti tak môžu segmentovať svoju sieť na menšie, bezpečnejšie zóny a chrániť svoje aplikácie a služby pred hrozbami a útokmi. Okrem toho NSX-T poskytuje širokú škálu bezpečnostných funkcií, ako sú distribuovaný firewall, distribuovaná detekcia a prevencia prienikov, ochrana pred DDoS útokmi a pokročilá prevencia hrozieb [32].

4.2.1 Architektúra NSX-T

NSX-T funguje prostredníctvom troch samostatných, ale integrovaných rovín, a to spravovacej, riadiacej a dátovej. Tieto tri roviny sú implementované ako sady pro-

cesov, modulov a agentov nachádzajúcich sa na dvoch typoch uzlov: manažérskom uzle a transportnom uzle (tieto pojmy budú špecifikované nižšie) [34].

Spravovacia rovina

Spravovacia rovina poskytuje vstupný bod do NSX-T na vykonávanie rôznych úloh ako je konfigurácia a monitorovanie všetkých troch rovín v architektúre NSX-T. Zmeny sú vykonávané prostredníctvom grafického rozhrania NSX-T Data Center alebo RESTful API.

Od verzie NSX-T 2.4 sa spravovacia a riadiaca rovina nachádza v NSX management clusteri, ktorý obsahuje tri NSX manažérske zariadenia v podobe virtuálnych strojov. Tým, že tento cluster obsahuje tri manažérske zariadenia je dosiahnutá vysoká dostupnosť (high availability).

Riadiaca rovina

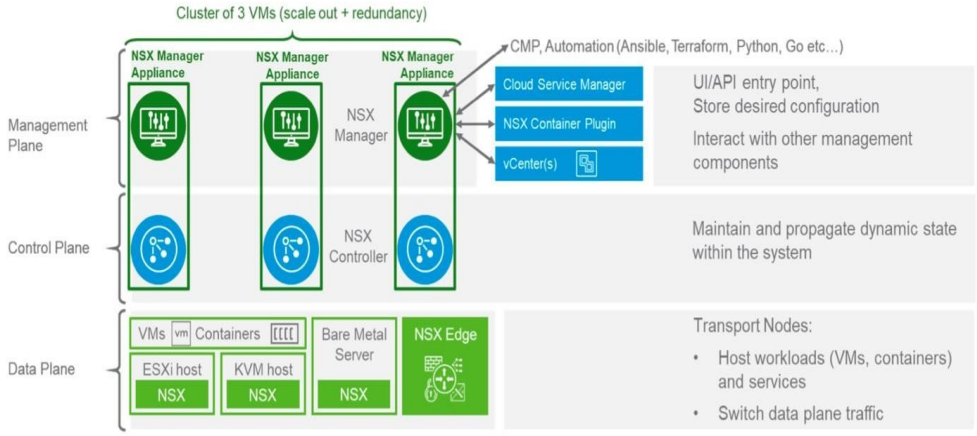
Úlohou riadiacej roviny je spravovať funkcie ako logické prepínanie a smerovanie alebo propagovať pravidlá distribuovaného firewallu. Ako už bolo uvedené, od verzie NSX-T 2.4 je riadiaca rovina spojená so spravovacou rovinou. Radiaca rovina v NSX-T je rozdelená do dvoch častí a to: centralizovanej riadiacej roviny a lokálnej riadiacej roviny.

- Centrálna riadiaca rovina (CCP) je implementovaná v NSX manažérskom zariadení. Jej úlohou je udržiavať globálny pohľad na sieť a ukladať informácie z lokálnej riadiacej roviny o logických prepínačoch, smerovačoch a ďalších sieťových a bezpečnostných nastaveniach. CCP je logicky oddelená od dátovej roviny, takže jej výpadok neovplyvní prebiehajúce procesy v dátovej rovine.

- Lokálna riadiaca rovina (LCP) sa nachádza v transportných uzloch. Úlohou LCP je riadiť činnosti dátovej roviny podľa inštrukcií z CCP.

Dátová rovina

Dátová rovina vykonáva odosielanie paketov na základe tabuliek z riadiacej roviny. Dátová rovina tiež hlási informácie o topológií siete riadiacej rovine. Obsahuje transportné uzly, ktoré sa delia na tri typy: hypervisor, edge a bare-metal transportný uzol. Hypervisor uzol obsahuje napr. ESXi a KVM hostov, ktorých súčasťou sú virtuálne zariadenia (VM), kontajnery a aplikácie. Bare-metal uzol referuje k hardvérovému zariadeniu, na ktorom nie je nainštalovaný hypervisor. Aplikácie na bare-metal serveri sú spustené priamo nad operačným systémom serveru. Pojem Edge transportný uzol bude uvedení nižšie.



Obr. 4.2: Architektúra NSX-T [34]

4.2.2 Komponenty NSX-T

Pred uvedením, ako funguje prepínanie či smerovanie v NSX-T, definuje táto sekcia niekoľko dôležitých súčastí a pojmov v prostredí NSX-T.

Transportný uzol (transport node)

Transportné uzly tvoria základ dátovej roviny a sú zodpovedné za prenášanie dátového toku smerujúceho a pochádzajúceho z virtuálnych počítačov alebo kontajnerov nachádzajúcich sa v systéme NSX-T. Ako už bolo spomenuté, existujú tri druhy hostov, ktoré tvoria transportný uzol: bare-metal host, hypervizor host a edge node. Každý transportný uzol obsahuje instanciu distribuovaného prepínača tzv. NSX-T Virtual Distributed Switch (N-VDS). Úlohou tohto distribuovaného prepínača je zabezpečiť konektivitu v rámci vnútorných komponent transporného uzla (virtuálne počítače, kontajnery) a fyzickou sieťou. V prípade použitia ESXi hosta je implementácia N-VDS odvodená od vDS. Jadro (kernel) transportného uzla okrem N-VDS tiež obsahuje LCP a MPA (Management Plane Agent), ktorý primárne slúži na zbieranie štatistík [35].

Transportná zóna (transport zone)

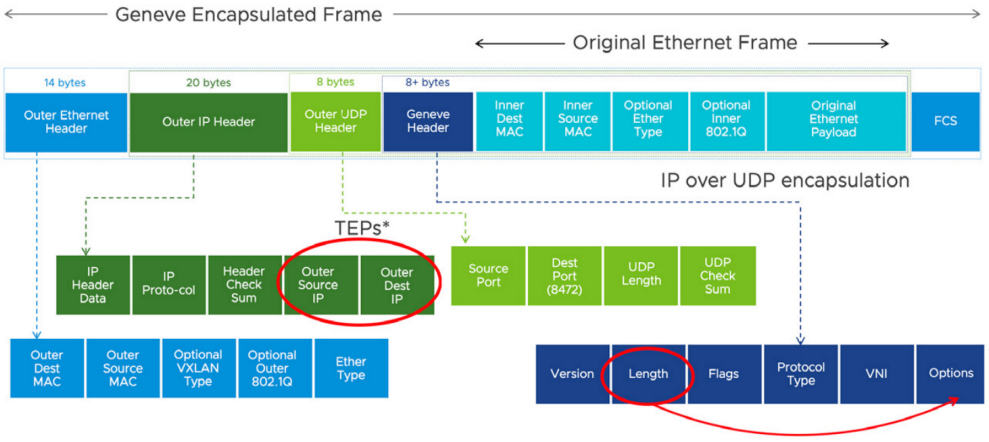
Transportná zóna je definovaná ako množina transportných uzlov, ktorá medzi sebou komunikuje skrz fyzickú infraštruktúru. Transportné uzly zdieľajú v transportnej zóne rovnakú instanciu N-VDS. Ten sa priraďuje ku transportnej zóne pri jej konfigurovaní. Toto prepojenie určuje, ktorý N-VDS zodpovedá za fungovanie siete v rámci danej transportnej zóny. Po pridaní transportného uzla do transportnej zóny sa na transportný uzol nainštaluje instancia N-VDS príslušnej

transportnej zóny. Transportný uzol môže byť súčasťou viacerých transportných zón. V NSX-T sa transportné zóny delia na dva typy [35]:

- Overlay transportná zóna - táto transportná zóna je využívaná hypervisor, bare-metal ako aj edge transportnými uzlami. Slúži na vytváranie overlay sietí a tunelov medzi transportnými uzlami v rámci infraštruktúry NSX-T.
- VLAN transportná zóna - je primárne určená na prepojenie edge transportných uzlov so sieťou mimo domény spravovanej NSX-T (napr. s fyzickou sieťou mimo dosahu NSX-T).

Tunnel End Point (TEP)

Každý transportný uzol obsahuje Tunnel End Point, čo je IP adresa identifikujúca jednu stranu overlay tunela. Zdrojová a cieľová IP adresa TEP v externej IP hlavičke paketu určuje, z ktorého transportného uzlu pochádza daný paket, a ku ktorému transportnému uzlu smeruje. Na tunelovanie je použitý protokol GENEVE. Pôvodný ethernetový rámec GENEVE zapúzdruje do UDP paketu, kde za L4 UDP hlavičku pridáva dodatočnú GENEVE hlavičku. V nej je dôležité pole VNI, ktoré identifikuje sieťový segment (virtuálna broadcastová doména). Analógiou ku VNI číslu je VLAN ID v 802.1Q hlavičke [35].



Obr. 4.3: UDP paket s GENEVE hlavičkou a polia GENEVE hlavičky [35]

Edge transportný uzol (edge transport node)

Edge transportný uzol je dedikovaný uzol poskytujúci sieťové služby (označované tiež ako aj servisné služby), ktoré nemožno distribuovať medzi hypervisor transportné uzly. Medzi ne patrí napr. NAT, L2 bridging, DHCP, alebo load-balancing.

Edge uzly tiež zabezpečujú konektivitu sever-juh so sieťami mimo prostredia NSX-T. Edge uzly buď majú podobu virtuálnych zariadení alebo bare-metal serveru. [35].

Edge cluster

Množina edge uzlov vytvára edge cluster. Jeho úlohou je zaručiť vysokú dostupnosť servisných služieb v prípade výpadku edge uzla [35].

4.2.3 Prepínanie v NSX-T

V rámci N-VDS v NSX-T je možné konfigurovať segmenty. Segment (alebo aj logický prepínač) predstavuje virtuálnu broadcastovú doménu, ktorá replikuje funkcie prepínania v prostredí NSX-T. Segmenty tak zabezpečujú L2 konektivitu medzi transportnými uzlami v transportnej zóne. V NSX-T existujú dva druhy segmentov: overlay segment a VLAN segment (podľa priradenej transportnej zóny). Transportný uzol môže byť súčasťou overlay alebo VLAN segmentu či prípadne oboch naraz. Segment môže byť súčasťou iba jednej transportnej zóny [36].

Overlay segment

Overlay segment vytvára logickú L2 sieť nad existujúcou fyzickou sieťou. NSX-T vytvára overlay segmenty tým, že tuneluje komunikáciu medzi dvoma virtuálnymi zariadeniami (prípadne komunikáciu medzi virtuálnym zariadením a gateway) za použitia GENEVE tunelovacieho protokolu. Ako už bolo spomenuté, každý transportný uzol obsahuje TEP s priradenou IP adresou. TEP tvoria koncové body medzi GENEVE tulenom skrz fyzickú sieť. GENEVE tunely sú ovládané NSX-T kontrolérom, čiže používanie GENEVE protokolu nevyžaduje dodatočnú konfiguráciu. Pri vytvorení overlay segmentu NSX-T kontrolér distribuuje instanciu logického prepínača do transportných uzlov v rámci danej transportnej zóny. Každému overlay segmentu je pridelené unikátne číslo VNI, ktoré identifikuje segment bez nutnosti použia VLAN tagov. Overlay segmenty taktiež fungujú bez nutnosti použitia protokolu STP [36].

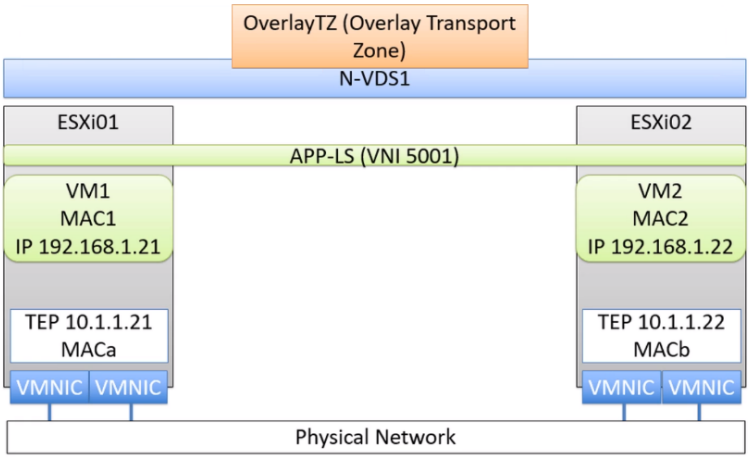
NSX-T kontrolér poskytuje transportným uzlom sadu troch tabuliek pre vykonávanie funkcií prepínania: ARP, TEP a MAC tabuľku [38].

- ARP tabuľka - obsahuje lokálne (v rámci segmentu) mapovanie medzi MAC adresou virtuálneho zariadenia a IP adresou virtuálneho zariadenia. Vďaka použitiu ARP tabuľky je možné potlačiť broadcastové správy z ARP protokolu. Ak odošle virtuálne zariadenie ARP request správu na zistenie MAC adresy cieľového zariadenia, odpovie na ňu logický prepínač s príslušným zá-

znamom z ARP tabuľky. Logický prepínač tak ďalej neodosiela ARP request od virtuálneho zariadenia.

- TEP tabuľka - vytvára mapovanie medzi TEP IP adresou a identifikátorom segmentu (VNI). Tabuľka tak obsahuje informácie, ktoré TEP IP adresy spadajú pod príslušný segment.
- MAC tabuľka - obsahuje mapovanie medzi MAC adresami virtuálneho zariadenia a IP adresou TEP. Určuje tak, ktoré MAC adresy spadajú pod daný TEP.

Na obrázku 4.4 je zobrazená jednoduchá sieť, ktorá pozostáva z dvoch ESXi hostov, dvoch virtuálnych zariadení a overlay segmentu. Oba transportné uzly sú súčasťou rovnakej overlay transportnej zóny. Virtuálne zariadenia sú pripojené na rovnaký segment (logický prepínač) s VNI číslom 5001. Pri komunikácií medzi zariadeniami VM1 a VM2 príjme tento logický prepínač rámec od VM1. Logický prepínač ďalej analyzuje cieľovú MAC adresu rámca a podľa MAC tabuľky zistí, že cieľová MAC adresa nepatrí lokálnemu TEP1. NSX-T kontrolér následne zostaví medzi zdrojovým TEP1 a cieľovým TEP2 GENEVE tunel skrz fyzickú sieť. TEP2 príjme zabalený rámec od VM1 a podľa VNI čísla identifikuje, ku ktorému segmentu patrí zabalený rámec. Následne logický prepínač s VNI číslom 5001 odošle originálny rámec VM2.



Obr. 4.4: Prepínanie v NSX-T

VLAN segment

VLAN segment taktiež vytvára virtuálnu broadcastovú doménu, ktorá je ale implementovaná ako tradičná VLAN vo fyzickej sieti. To vyžaduje, aby dátový tok medzi

virtuálnymi zariadeniami na rôznych transportných uzloch v rámci segmentu bol označený napr. 802.1Q tagom. Danú VLAN je potom ale ešte nutné donastaviť na fyzickej sieti. Zvyčajne VLAN segmenty prepájajú Edge transportné uzly, ktoré tiež zaisťujú konektivitu mimo prostredia NSX-T [37].

4.2.4 Smerovanie v NSX-T

Platforma NSX-T Datacenter umožňuje vytvárať virtuálne L3 siete za použitia logických smerovačov. Tie majú softvérovú podobu a sú distribuované v N-VDS prepínači v transportných uzloch. Implementovaním logického smerovania možno poskytnúť konektivitu k externým sieťam mimo prostredie NSX-T alebo v rámci NSX-T pri smerovaní dátového toku medzi L2 segmentmi [36].

Typy smerovačov

Logické smerovanie v NSX-T sa vykonáva pomocou dvoch typov smerovačov: Tier-0 a Tier-1. Tier-0 smerovač sa považuje za smerovač poskytovateľa služieb a Tier-1 smerovač funguje primárne ako smerovač nájomcu (tenant) [36].

Tier-0 smerovač poskytuje pomocou uplink rozhraní pripojenie k externým sieťam. Smerovanie možno nastaviť buď za použitia statických ciest alebo použitím protokolu BGP. Ku downlink rozhraniam Tier-0 smerovača možno pripájať L2 segmenty ako aj viacero Tier-1 smerovačov. Úlohou Tier-0 smerovača je vymieňať smerovacie informácie s externými sieťami a poskytovať konektivitu medzi Tier-1 smerovačmi. Tier-1 smerovače sa pripájajú svojimi uplinkovými rozhraniami k Tier-0 smerovačom a tiež predstavujú bod, ku ktorému sa pripájajú L2 segmenty [36].

Ak sú v sieti prítomné smerovače Tier-0 aj Tier-1, Tier-0 má na starosti predovšetkým smerovanie dátového toku typu sever-juh medzi virtualizovanou NSX-T sieťou a vonkajšou fyzickou sieťou. Tier-1 smerovač je optimalizovaný na prenos dátového toku v smere východ-západ (medzi L2 segmentmi) [36].

Distribuovaný a servisný smerovač

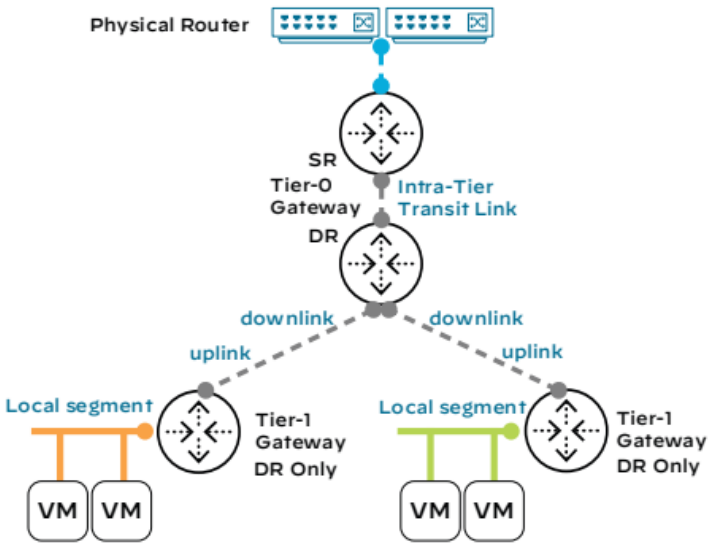
Logické Tier-0 a Tier-1 smerovače pozostávajú z dvoch odlišných komponetov a to: distribuovaného smerovača (DR) a servisného smerovača (SR). DR komponent logického smerovača má podobu modulu v jadre N-VDS, čím umožňuje NSX-T distribuovať funkcionality smerovania do transportných uzlov (hypervisor aj edge uzly). DR umožňuje každému transportnému uzlu lokálne smerovať dátový tok sever-juh a východ-západ priamo na N-VDS. Tier-0 a Tier-1 smerovače majú vždy DR komponent, ale nemusia mať SR komponentu [36].

SR je komponent Tier-0 alebo Tier-1 smerovača, ktorý nemožno distribuovať

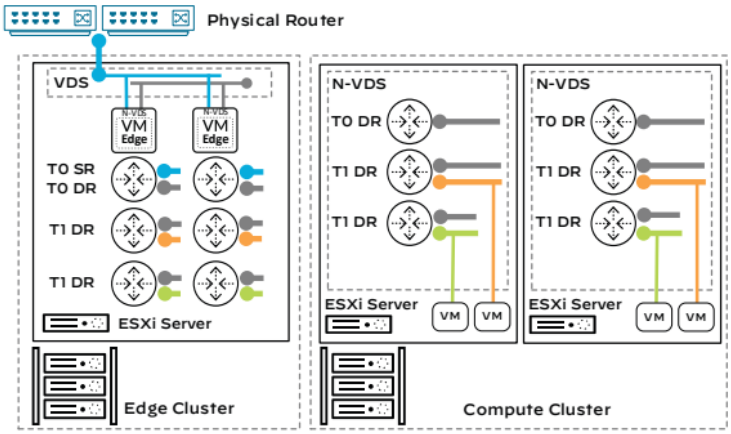
medzi transportnými uzlami. Nachádza sa v Edge uzle a obsahuje služby Tier-0 a Tier-1 smerovača ako napr. NAT, pripojenie k externej sieti alebo load-balancing. Akýkoľvek dátový tok, ktorý potrebuje použiť niektorú zo servisných služieb je preposielaný do Edge uzlu na spracovanie [36].

Pri nakonfigurovaní Tier-1 alebo Tier-0 smerovača sa nasadí DR komponent smerovača do všetkých transportných uzlov. Pri vytvorení niektorej z centralizovaných služieb je SR komponent smerovača nasadený na Edge uzli v Edge clusteri. Na prepojenie SR a DR komponentov slúži tzv. intra-tier transit link. Jedná sa o L2 segment, ktorý je automaticky vytváraný a spravovaný NSX-T kontrolérom.

Príklad jednoduchej logickej siete v NSX-T, ktorá pozostáva z dvoch segmentov, dvoch Tier-1 smerovačov a jedného Tier-0 smerovača je zobrazená na obrázku 4.5. Následne obrázok 4.6 zobrazuje fyzickú reprezentáciu tejto siete na transportných uzloch.



Obr. 4.5: Reprezentácia logickej siete v NSX-T [36]



Obr. 4.6: Reprezentácia fyzickej siete v NSX-T [36]

Dvojúrovňové smerovanie

NSX-T podporuje viacúrovňový model smerovania, kde sú logicky oddelené funkcie pre smerovač poskytovateľa a smerovač nájomcu. Táto štruktúra poskytuje správcom siete a nájomcom ovládať svoje vlastné procesy a služby. V takejto topológií siete spravuje Tier-0 smerovač poskytovateľ a Tier-1 smerovač nájomca [36].

V prostredí NSX-T možno použiť aj jednoúrovňové smerovanie implementované iba pomocou Tier-0 smerovačov. Použitie viacúrovňového smerovania oproti jednoúrovňovému má ale množstvo výhod. Jednou z nich je nezávislosť na konfigurácií fyzickej siete. Pri pridaní nového Tier-1 smerovača nájomcu Tier-0 logický smerovač iba oznámi fyzickej sieti nové cesty, ktoré sa Tier-0 smerovač naučil od Tier-1 smerovaču [36].

Koncept distribuovaného a servisného smerovača ostáva v dvojvrstvovej či jednovrstvovej architektúre rovnaký.

Medzi Tier-0 a Tier-1 logickými smerovačmi neprebieha dynamické smerovanie. Platforma NSX-T zabezpečuje smerovanie medzi Tier-0 a Tier-1 vrstvou smerovačov automaticky. Nasledujúci zoznam popisuje niektoré druhy ciest u Tier-0 a Tier-1 smerovačov [35]:

- **Tier-1**
  - Statická cesta - je manuálne nastavená cesta cez NSX-T Manager.
  - NSX Connected – je priamo pripojená cesta k rozhraniu smerovača.
- **Tier-0**
  - Statická cesta - je manuálne nastavená cesta cez NSX-T Manager.

- NSX Connected – je priamo pripojená cesta k rozhraniu smerovača.
- NSX Static - sú statické cesty, ktoré vytvára automaticky NSX-T Manager pre zaistenie konektivity medzi Tier-0 smerovačom a segmentami pripojenými na rozhrania Tier-1 smerovača.
- BGP - sú cesty naučené z protokolu BGP.

4.2.5 Bezpečnosť v NSX-T

Prostredie NSX-T prináša novú paradigmu firewallovej stratégie. V tradičných sieťach hardvérový firewall predstavuje centrálny bod v sieti, na ktorý smeruje celý dátový tok. V NSX-T majú softvérové firewally distribuovaný charakter, a preto ich možno nasadiť priamo na virtuálnych zariadeniach. Existujú tri typy firewallov v NSX-T: Gateway firewall, Distributed firewall a Bridge firewall [39].

Distribuovaný firewall (DFW) je typ firewallu, ktorý slúži na monitorovanie a zabezpečenie dátového toku v smere východ-západ. DFW možno nasadiť priamo na virtuálnu sieťovú kartu (vNIC), čo umožňuje vykonávať inšpekciu paketov smerujúcu do a mimo virtuálneho zariadenia. Tieto DFW fungujú ako modul v jadre hypervisora. Pri migrácií virtuálneho zariadenia taktiež migrujú aj nastavené pravidlá na DFW. DFW sa konfiguruje centrálne ako jeden univerzálny firewall a zmeny v nastavení pravidiel firewallu sa následne distribuujú do transportných uzlov [39].

Gateway firewall (GFW) sprístupňuje bezpečnostné funkcionality na hranici (perimetri) siete a tým zabezpečuje dátový tok v smere sever-juh. Tieto funkcionality zahrňujú služby ako stateful alebo stateless inšpekcia paketov či VPN. GFW možno nasadiť na rozhrania Tier-0 ako aj na Tier-1 smerovačov, ale samotný GFW je umiestnený v Edge transportnom uzle. Typicky GFW zabezpečujú dátový tok medzi dvoma fyzickými servermi, virtuálnym a fyzickým serverom či v multi-tenant prostrediach. Firewally GFW a DFW fungujú nezávisle na sebe a každý druh firewallu udržiava svoju vlastnú konfiguráciu.

Bridge firewall je používaný iba medzi NSX bridge. NSX bridge slúži na spájanie segmentov v rámci druhej vrstvy OSI modelu (bez smerovania) [39].

Kapitola 5

Automatizácia procesov

Táto kapitola sa zaoberá popisom vytvoreného riešenia na automatizáciu spravovania sieťových segmentov v prostredí NSX-T a virtuálnych zariadení vo vCenter. Kapitola tiež popisuje použité nástroje Jenkins a Terraform vo výslednom riešení. Záver kapitoly obsahuje diskusiu o vytvorenom riešení, jeho možných úpravách a nadstavbách, a tiež jeho bezpečnostné aspekty.

Nástroje vCenter a NSX-T možno riadiť pomocou API bez nutnosti používania užívateľského rozhrania. Cieľom v Algotechu je presunúť niektoré rutinné úlohy ako vytváranie sieťových segmentov a virtuálnych zariadení operátorom Service desku. Zamestnanci Service desku, ktorí nutne nemusia mať znalosti o konfigurovaní vCenter a NSX-T, by napr. prostredníctvom externého webového rozhrania mohli zadávať základné parametre na spravovanie segmentov alebo virtuálnych zariadení. Cieľom v praktickej časti diplomovej práce je simulovať takéto užívateľské rozhranie a automatizovať potrebné medzikroky, ktoré by požadované parametre zadané z tohto rozhrania transformovali na zmenu v konfigurácií NSX-T alebo vCenter. Zadanie praktickej časti diplomovej práce teda obsahuje dva body:

- Automatizovať proces spravovania L2 segmentov v prostredí NSX-T.
- Automatizovať proces spravovania virtuálnych zariadení (VM) vo vCenter.

5.1 Použité nástroje

V riešení zadania praktickej časti diplomovej práce boli použité nástroje Jenkins a Terraform. V práci boli tiež implementované dodatočné skripty v jazyku Python. Subsekcia nižšie v krátkosti opisuje oba zmienené nástroje a ich funkcionality.

5.1.1 Jenkins

Ako automatizačný nástroj, ktorý spúšťa skripty a riadi potrebné medzikroky pri prenose konfigurácie ku NSX-T alebo vCenter, bol zvolený Jenkins. Jenkins je open-source automatizačný server, ktorý poskytuje sadu doplnkov na automatizáciu rôznych aspektov softvérového vývoja. Jenkins ponúka množstvo rôznych nástrojov, ktoré pomáhajú pri vytváraní pipeline a nasadzovaní programov. Jenkins umožňuje vytvárať tzv. úlohy, kde každá úloha (job) predstavuje vývojový cyklus od zostavenia až po nasadenie programu v cieľovom prostredí. Tieto Jenkins úlohy môžu byť nastavené tak, aby sa spúšťali automaticky pri určitých udalostiach, ako napr. pri detekovaní zmeny v zdrojovom kóde alebo v pravidelných časových intervaloch. Samotný Jenkins je napísaný v Java programovacom jazyku a jednotlivé úlohy možno v Jenkins vytvárať prostredníctvom Groovy programovacieho jazyka [40].

Jednou z výhod Jenkins je jeho schopnosť integrovať sa s rôznymi nástrojmi pomocou doplnkov a pluginov, ktoré umožňujú rozšíriť jeho funkcionalitu. Jenkins je tak efektívnym nástrojom pre automatizáciu opakujúcich sa úloh, čím šetrí vývojárom čas a zvyšuje celkovú efektivitu vývojového procesu [40].

Okrem toho Jenkins poskytuje užívateľom prehľadné rozhranie s informáciami o priebehu jednotlivých úloh. To umožňuje vývojárom jednoducho zobrazovať výsledky testovania, chyby a stav jednotlivých krokov v Jenkins úlohe.

Jenkins úloha pozostáva z krokov (stage), ktoré sú medzi sebou zreťazené (tvoria pipeline). Pri chybe v niektorom z krokov je úloha prerušená.

V rámci diplomovej práce bolo v prostredí Jenkins napísaných niekoľko úloh pre spravovanie segmentov v NSX-T a virtuálnych zariadení vo vCenter. Úlohy sú spúšťané manuálne (napr. operátorom service desku) a vykonávajú určitú operáciu v cieľovom prostredí (napr. pridanie segmentu v NSX-T).

Jenkins umožňuje pred spustením úloch ich parametrizáciu. To umožňuje pridať do Jenkins úlohy vstupné premenné, ktoré ovlyvňujú následný priebeh krokov v Jenkins úlohe. Vstupný formulár v Jenkinse na parametrizáciu úlohy simuluje užívateľské rozhranie, do ktorého administrátor zadáva potrebné parametre na vykonanie úlohy v cieľovom prostredí.

V diplomovej práci slúži prostredie Jenkins na zadávanie vstupných parametrov a na následné automatické spúšťanie doprovodných skriptov napísaných v jazyku Python. Úlohou Jenkins je taktiež spúšťať Terraform server pre vykonávanie konfiguračných zmien v prostrediach NSX-T a vCenter.

5.1.2 Terraform

Terraform je open-source nástroj od firmy Hashicorp používaný na automatizáciu správy infraštruktúry. Hlavnou úlohou Terraformu je umožniť vývojárom a admi-

nistrátorom deklaratívne popísať infraštruktúru prostredia v kóde a následne ju spravovať pomocou konfiguračných súborov [41].

Terraform konfiguračný súbor popisuje želaný stav infraštruktúry. Podľa zdrojového kódu Terraform následne vytvára, aktualizuje alebo odstráňuje virtuálne instancie v cieľovom prostredí. Terraform je tak veľmi flexibilný nástroj, ktorý umožňuje automatizovať správu infraštruktúry naprieč rôznymi prostrediami.

Sučasťou Terraformu je aj tzv. poskytovateľ (provider). Jedná sa o plugin, ktorý umožňuje Terraformu interagovať s API alebo službami daného prostredia. To môže zahŕňať cloudových poskytovateľov ako AWS, Azure, ale aj iné nástroje a technológie, ako napr. Kubernetes. Provider tak umožňuje Terraformu pracovať prakticky s akoukoľvek platformou alebo službou, ktorá disponuje API rozhraním [41].

Ukážka 5.1 zobrazuje zdrojový kód s definovaným providerom pre NSX-T, ktorý obsahuje IP adresu NSX-T serveru, prihlasovacie údaje a možnosť ignorovať autentizáciu prostredníctvom SSL certifikátu.

```
terraform {
    required_providers {
        nsxt = {
            source = "vmware/nsxt"
        }
    }
}

provider "nsxt" {
    host                  = "var.nsx_ip_address"
    username              = "var.nsx_username"
    password              = "var.nsx_password"
    allow_unverified_ssl = true
}
```

Zdrojový kód 5.1: Ukážka kódu s Terraform providerom

5.2 Schéma riešenia

Všetky vytvorené Jenkins úlohy na spravovanie virtuálnej infraštruktúri zdieľajú podobnú architektúru. Procesy, ktoré prebiehajú v Jenkins úlohe možno popísať do piatich krokov:

1. V prvom kroku Jenkins pipeline administrátor zadá do zadávacieho formuláru potrebné údaje pre vykonanie nejakej činnosti (napr. vytvorenie virtuálneho zariadenia). Po zadaní vstupných parametrov užívateľ spustí Jenkins úlohu, ktorá následne automaticky spustí pipeline na prenos zadanej konfigurácie do cieľového prostredia (napr. NSX-T)

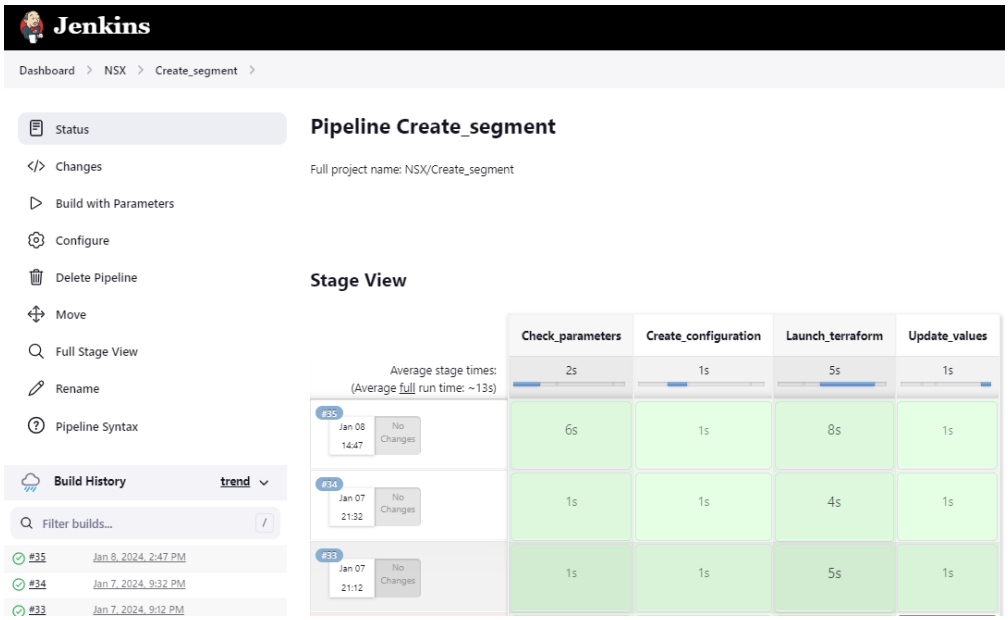
2. Druhým krokom je skontrolovanie zadaných údajov prostredníctvom Python skriptu, ktorý overuje, či užívateľ zadal validné vstupy do Jenkins formuláru. V prípade nájdenia chyby sa pipeline preruší a upozorní užívateľa na vadný vstup vo formulári. V rámci tohto kroku prebieha tiež kontrola či nová konfigurácia nenarušuje súčasné nastavenie cieľového prostredia (napr. či nový segment v NSX-T nemá duplicitnú IP adresu s existujúcimi segmentmi).

3. Ak boli vstupné parametre zadané správne, Jenkins spustí ďalší Python skript, ktorý podľa požadovanej akcie vytvorí, aktualizuje alebo vymaže zdrojový kód virtuálnej instancie v Terraform konfiguračnom súbore.

4. Vo štvrtom kroku Jenkins inicializuje Terraform server. Ten prečíta aktualizovaný konfiguračný súbor, napojí sa skrz API na API cieľového prostredia a aplikuje požadované zmeny.

5. V poslednom kroku pipeline Jenkins spustí Python skript, ktorý aktualizuje textové súbory potrebné pre výber údajov v zadávacom formulári.

Schému pipeline zmienenej vyššie ako aj užívateľské rozhranie Jenkins zobrazuje obrázok 5.1.



Obr. 5.1: Užívateľské rozhranie Jenkins

Výsledné riešenie sa tak primárne opiera o nástroje Jenkins a Terraform. Doplňujúce Python skripty slúžia na implementovanie dodatočnej logiky pri kontrolovaní vstupných údajov, upravovaní Terraform konfiguračného súboru a zisťovanie

aktuálneho stavu prostredí NSX-T a vCenter. Jazyk Python bol použitý najmä pre jeho dostupné knižnice na vytváranie HTTP GET requestov a následne spracovanie prijatých odpovedí z API od NSX-T a vCenter v JSON formáte. Opisom samotných Python skriptov sa diplomová práca nevenuje, keďže implementujú elementárnu kontrolu vstupných parametrov a formátovanie a úpravu textovej šablóny. Po inštalácií Python a Terraform nie je nutné dodatočne sťahovať nijaký plugin alebo knižniče. V prípade Jenkins pre fungovanie výsledného riešenia je ešte nutné doinštalovať plugin Credentials a Extended Choice Parameter Plugin.

Pred prvotným použitím Jenkins úloh na správu virtuálnych instancií je ešte nutné spustiť dedikované Jenkins úlohy (Prepare working directory) na prípravu pracovného adresára. Ten po dokončení úlohy obsahuje predpripravené súbory pre Jenkins a Terraform konfiguračné súbory. Taktiež pred spustením úloh na správu virtuálnej infraštruktúry je nutné zadať potrebné prihlasovacie údaje oboch cieľových prostredí do Jenkins.

V zadávacom Jenkins formulári možno zadať niekoľko dátových typov vstupných premenných. V diplomovej práci boli najmä použité nasledujúce typy vstupných premenných:

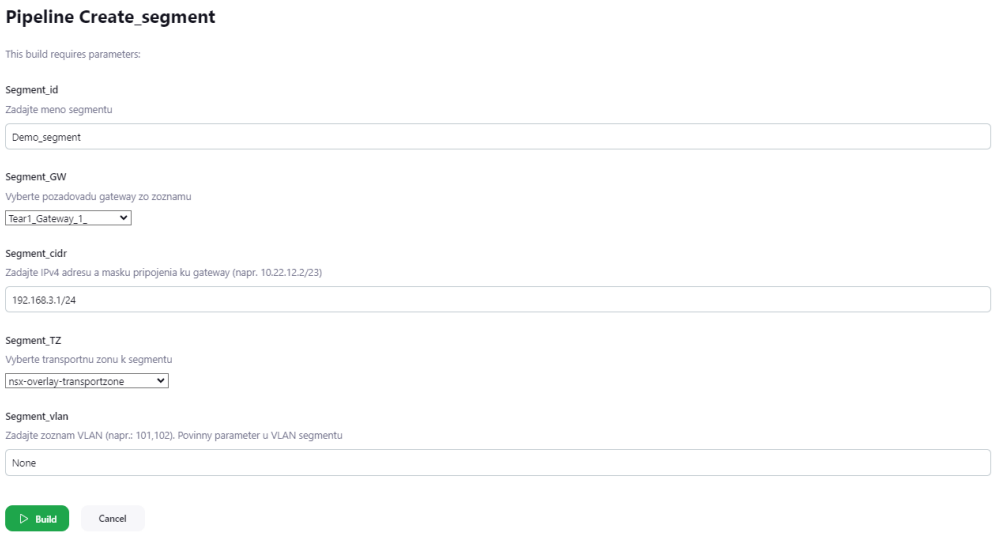
- String parameter - jedná sa o klasickú premennú typu string, kde užívateľ zadá určitý text.
- Boolean parameter - je premenná, ktorá obsahuje logickú hodnotu True alebo False.
- Choice parameter - je premenná, ktorá obsahuje preddefinované hodnoty a užívateľ tak len vyberie jednu hodnotu z ponúkaných možností.

5.3 Vytváranie nového segmentu v NSX-T

Ako už bolo načrtnuté v predošlej subkapitole, v prvom kroku pred spustením Jenkins úlohy vyplní administrátor zadávací formulár o povinné údaje na vytvorenie segmentu. Zadávací formulár je zobrazený na obrázku 5.2 a obsahuje nasledujúce vstupné parametre:

- Segment_id - definuje meno segmentu, ktoré musí byť unikátne v NSX-T
- Segment_gw - obsahuje zoznam Tier-1 smerovačov, kde užívateľ vyberie, ku ktorému Tier-1 smerovaču pripojí nový segment.
- Segment_ip - parameter, ktorý vytvára na Tier-1 smerovači virtuálne rozhranie podľa zadanej IPv4 adresy a dĺžky prefixu. ID siete následne dopočíta NSX-T.

- Segment_transport_zone - slúži na výber transportnej zóny, ktorej súčasťou má byť nový segment.
- Segment_vlan - jedná sa o povinný parameter pri vytváraní VLAN segmentu, kde užívateľ zadá jednotlivé VLAN.



Obr. 5.2: Zadávací formulár pre vytvorenie segmentu

Po zadaní všetkých parametrov a spustení Jenkins úlohy prebehne kontrola vstupných parametrov. Ak užívateľ nezadal všetky povinné parametre resp. zadal nesprávnu hodnotu u niektorého z parametrov, dôjde k prerušeniu pipeline a celej Jenkins úlohy. V implementovanom Python skripte prebiehajú kontroly ako napr. či na príslušnom Tier-1 smerovači by nedošlo k vytvoreniu segmentu s duplicitnou IP adresou alebo či boli zadané validné čísla VLAN. Prerušenie Jenkins úlohy nastane aj v prípade, ak užívateľ zadal meno segmentu, ktoré už existuje v NSX-T.

Po overení správnosti vstupných údajov sú tieto parametre naformátované do Terraform konfiguračného súboru. Ukážka 5.2 obsahuje použitý kód s naformátovanými vstupmi, ktorý slúži na vytvorenie overlay segmentu.

```
resource "nsxt_policy_segment" "segment_test" {

    nsx_id              = "segment_test"
    display_name        = "segment_test"
    description         = "Terraform provisioned overlay segment"
    connectivity_path   = "var.tier_1_absolute_path"
    transport_zone_path = "var.transport_zone_path"
```

```
8
9       subnet {
10          cidr          = "192.168.0.1/24"
11      }
12  }
```

Zdrojový kód 5.2: Ukážka kódu s konfiguráciou segmentu

Po doplnení Terraform konfiguračného súboru o nastavenie nového segmentu prebehne spustenie Terraform serveru. Tu Jenkins spúšťa dva príkazy:

- Terraform init - príkaz, ktorý slúži na stiahnutie pluginov a pripravuje pracovný adresár s Terraform konfiguračnými súbormi.
- Terraform apply - príkaz, ktorý sa spojí s API cieľového prostredia a podľa konfiguračného súboru vykoná zmeny v infraštruktúre cieľového prostredia.

Terraform server uchováva svoju predošlú konfiguráciu a následne ju porovnáva s aktuálnou konfiguráciou. Takto Terraform zisťuje, aké zmeny v konfigurácií cieľového prostredia treba vykonať. Po dokončení príkazu Terraform apply je v NSX-T vytvorený nový segment.

Posledným krokom v Jenkins pipeline pre vytvorenie segmentu je aktualizovanie textových súborov, ktoré využíva Jenkins na zobrazovanie možností vo vstupnom formulári. Medzi tieto údaje patria názvy Tier-1 smerovačov, transportných zón či názvy segmentov vytvorených Terraformom. V rámci jedného cyklu Jenkins úlohy je možné vytvárať len jeden segment, hromadná konfigurácia viacerých segmentov nie je možná.

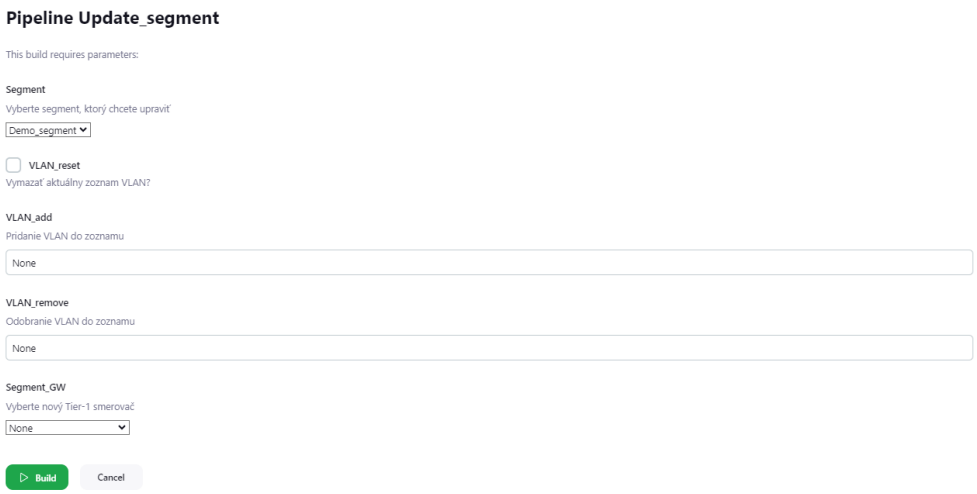
5.4 Úprava existujúceho segmentu v NSX-T

Segmenty, ktoré boli vytvorené prostredníctvom Terraformu možno dodatočne podľa potreby upravovať. Ak je potrebné zmeniť niektoré zo základných údajov ako napr. pripojený Tier-1 smerovač, nie je nutné vymazať segment a vytvoriť nový segment s požadovanou konfiguráciou. Prvý krok Jenkins úlohy obsahuje podobne ako v minulom prípade zadávací formulár. Zadávací formulár, ktorý je zobrazený na obrázku 5.3 pozostáva z nasledujúcich vstupných premenných:

- Segment - premenná slúži na výber niektorého z Terraform segmentov, ktorý potrebuje administrátor upraviť
- VLAN_reset - parameter, určený najmä pre VLAN segmenty, ktorý vymaže aktuálny zoznam VLAN.
- VLAN_add - slúži na pridanie dodatočných VLAN do aktuálneho zoznamu.

- VLAN_remove - parameter slúžiaci na odstránenie konkrétnych VLAN z aktuálneho zoznamu.

- Segment_GW - parameter slúžiaci na prepojenie segmentu k inému Tier-1 smerovaču.

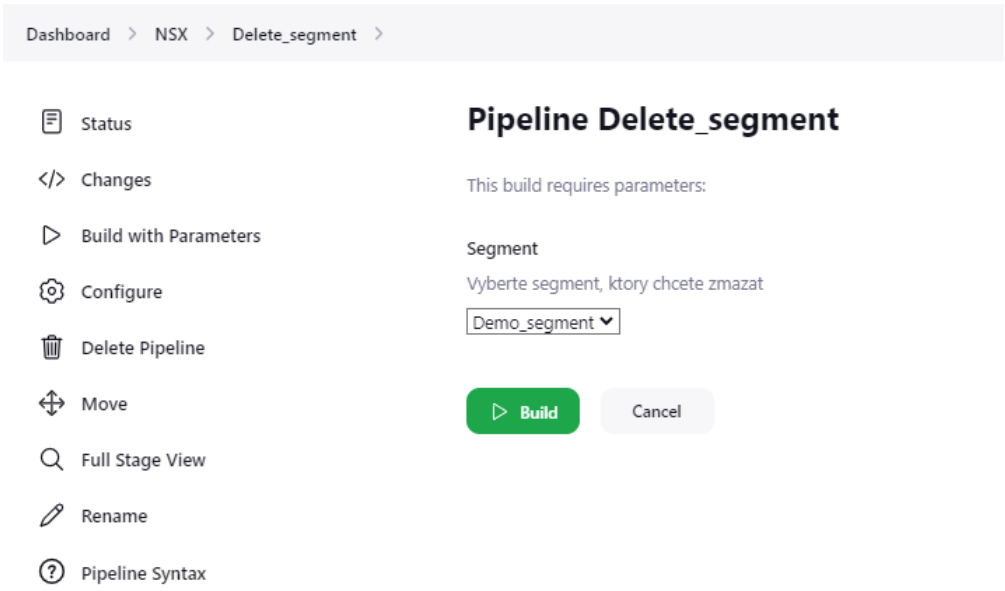


Obr. 5.3: Zadávací formulár pre úpravu segmentu

Parametre, ktoré neboli v zadávacom formulári vyplnené, ostávajú bez zmeny podľa pôvodnej konfigurácie. Implementované Python skripty ďalej skontrolujú zadané vstupné údaje zo zadávacieho formuláru a následne upravia pôvodné nastavenie Terraform konfiguračného súboru. Po spustení Terraform serveru sa aplikujú zmeny v NSX-T. Keďže nedošlo k vytvoreniu alebo zmazaniu segmentov nie je nutné aktualizovať textové súbory pre Jenkins.

5.5 Vymazanie segmentu v NSX-T

Táto Jenkins úloha slúži na zmazanie niektorého zo segmentu vytvoreného prostredníctvom Terraformu z NSX-T. Zadávací formulár zobrazený na obrázku x obsahuje jediný parameter, kde užívateľ vyberie, ktorý segment potrebuje vymazať. Z Terraform konfiguračného súboru sa odstráni časť zdrojového kódu definujúca daný segment a následné spustenie Terraformu vymaže segment z NSX-T. Obrázok 5.4 zobrazuje vstupný Jenkins formulár na vymazanie segmentu z NSX-T.



Obr. 5.4: Zadávací formulár pre vymazanie segmentu

5.6 Vytváranie nového virtuálneho zariadenia

Druhou časťou v zadaní praktickej časti diplomovej práce bolo vytvoriť Jenkins úlohy pre správu virtuálnych zariadení vo vCenter. Tieto Jenkins úlohy zdieľajú podobnú štruktúru aj analogický priebeh, ako úlohy pre správu segmentov v NSX-T, len cielia na iné prostredie.

Naskytli sa dve možnosti pri riešení problému vytvárania virtuálnych zariadení skrz Jenkins. Prvým riešením je vytvoriť virtuálne zariadenie vo vCenter prostredníctvom Jenkins, nahrať na jeho disk inštalačný súbor s požadovaným operačným systémom a následne v Jenkins implementovať jeho autoinštaláciu. Druhou možnosťou je vytvoriť šablónu virtuálneho zariadenia s vopred nainštalovaným operačným systémom priamo vo vCenter. Jenkins by následne pri vytváraní použil túto šablónu a naklonoval by podľa nej nové virtuálne zariadenie s pôvodným operačným systémom šablóny. V diplomovej práci bola použitá druhá možnosť pri vytváraní virtuálnych zariadení. Tá je účinná, ak nie je nutné manuálne vytvárať veľký počet šablón s rôznymi operačnými systémami. V rámci diplomovej práce bola vytvorená jedna šablóna s nainštalovanou linuxovou distribúciou CentOS 8.

Vytvorenie nového virtuálneho zariadenia obsahuje väčšie množstvo vstupných parametrov ako vytváranie segmentov v NSX-T. Jenkins zadávací formulár obsahuje nasledujúce vstupné premenné na vytvorenie nového virtuálneho zariadenia:

- VM_name - definuje meno nového virtuálneho zariadenia.
- Num_cpu - slúži na alokovanie vybraného počtu virtuálnych CPU pre virtuálne zariadenie.
- Size_ram - slúži na alokovanie vybranej veľkosti pamäti RAM.
- Size_hdd - slúži na vytvorenie HDD s požadovanou veľkosťou.
- VM_OS - je parameter, ktorý vyberá požadovaný operačný systém. Tento parameter určuje, podľa ktorej šablóny sa bude vytvárať virtuálne zariadenie.
- VM_folder - parameter, ktorý určuje cieľové umiestnenie virtuálneho zariadenia.
- Resource_pool - slúži pre vybranie niektorého z resource poolov, ktoré poskytujú výpočtové zdroje pre virtuálne zariadenie.
- Data_store - vyberá fyzické úložisko pre virtuálne zariadenie.
- Network - parameter, kde operátor vyberá, do ktorej siete by chcel pripojiť virtuálne zariadenie. Možno aj vybrať siete, ktoré boli vytvorené prostredníctvom Jenkins úlohy na vytváranie sietí.
- VM_ip_address - parameter určuje IPv4 adresu na virtuálnom rozhraní zariadenia.
- VM_prefix_length - parameter slúžiaci na zadanie dĺžky prefixu vybranej siete.
- VM_default_gw - parameter obsahujúci IPv4 adresu default gateway.
- VM_hostname - parameter, ktorý vytvára zadaného užívateľa v operačnom systéme virtuálneho zariadenia.
- VM_domain - slúži pre výber domény, ktorej súčasťou má byť vytvorené virtuálne zariadenie.

Po zadaní všetkých parametrov nasleduje ich kontrola. V nej sa napr. overuje či zariadenie neobsahuje duplicitnú IP adresu v rámci pripojeného segmentu alebo či zariadenie nemá alokované príliš veľké výpočtové zdroje. Ak sú všetky parametre potvrdené ako validné, nasleduje ďalší krok Jenkins pipeline v podobe formátovania šablóny pre vytvorenie virtuálneho zariadenia v Terraform. Ukážka 5.3 obsahuje zdrojový Terraform kód, do ktorého sa formátujú údaje zadané v zadávacom formulári.

```
resource "vsphere_virtual_machine" "vm_name" {
  name             = var.vm_name
  resource_pool_id = var.resource_pool_id
  datastore_id     = var.datastore_id
  num_cpus         = var.num_cpus
  memory           = var.size_ram
  guest_id         = var.guest_os
  firmware         = var.firmware
  folder           = var.folder
  network_interface {
    network_id   = var.selected_network
  }
  disk {
    label            = "disk0"
    size             = var.hdd_size
  }
  clone {
    template_uuid = var.template_uuid
    customize {
      linux_options {
        host_name = var.hostname
        domain    = var.domain
      }
      network_interface {
        ipv4_address = var.ipv4_address
        ipv4_netmask = var.prefix_length
      }
      ipv4_gateway = var.ipv4_gw
    }
  }
}
```

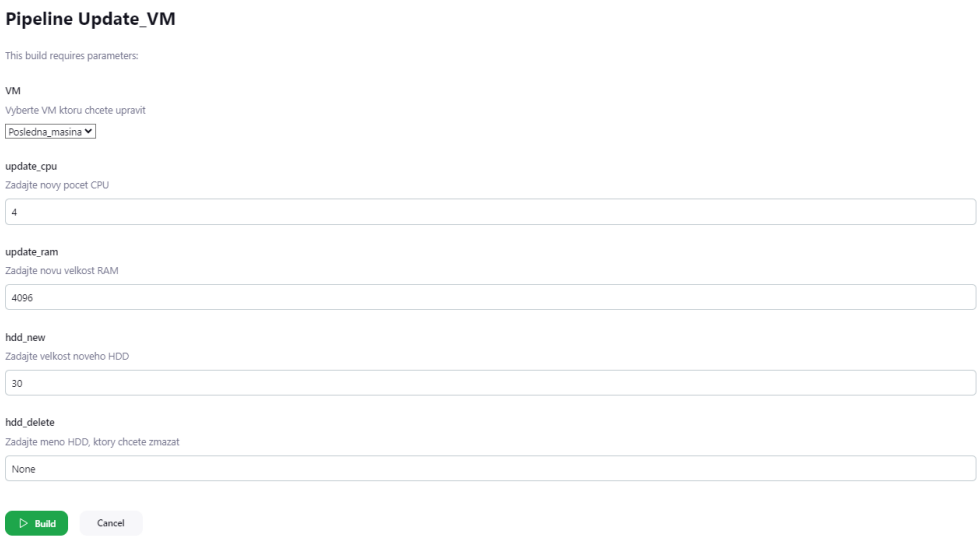
Zdrojový kód 5.3: Ukážka kódu so šablónou pre vytváranie virtuálneho zariadenia

Po naformátovaní a pridaní konfigurácie na vytvorenie nového virtuálneho zariadenia do Terraform konfiguračného súboru je spustený Terraform server. Ten sa následne spojí s API vCenter a vykoná potrebné zmeny. Nasadenie nového virtuálneho zariadenia cez Terraform trvá približne dve minúty. Po ukončení tejto periódy je už virtuálne zariadenie pripravené na použitie a napojené do požadovanej virtuálnej siete. Iný užívateľ tak dokáže ďalej nastavovať toto virtuálne zariadenie prostredníctvom vzdialeného prístupu. Posledným krokom v pipeline pre vytvorenie nového virtuálneho zariadenia je len aktualizácia súborov, ktoré používa Jenkins pri zobrazovaní možností vo vstupnom formulári.

5.7 Úprava existujúceho virtuálneho zariadenia

Vytvoreným virtuálnym zariadeniam prostredníctvom Terraformu možno dodatočne upravovať virtuálne hardvérové zdroje. V zadávacom formulári uvedenom na obrázku 5.5 je možné prenastaviť nasledujúce parametre:

- VM - parameter, kde užívateľ vyberie, ktoré virtuálne zariadenie by chcel upravovať.
- update_cpu - slúži na prenastavenie počtu virtuálnych CPU u zariadenia.
- update_ram - slúži na prenastavenie veľkosti alokovanej pamäti RAM.
- hdd_new - pridá dodatočné úložisko s definovanou kapacitou.
- hdd_delete - slúži na zadanie mena pevného disku, ktorý užívateľ chce zmazať.



Obr. 5.5: Zadávací formulár pre úpravu virtuálneho zariadenia

Užívateľ nutne nemusí vyplniť každý jeden z parametrov, ak iba napr. potrebuje upraviť veľkosť pamäti RAM. Nezadané parametre v zadávacom formulári sú ignorované a je u nich ponechané pôvodné nastavenie. Po zadaní nových parametrov prebehne opäť ich kontrola. Ak boli zadané parametre validné, nasleduje prepis konfigurácie zdrojového kódu u príslušného virtuálneho zariadenia. Konečným krokom v pipeline pre úpravu parametrov niektorého z virtuálnych zariadení je spustenie Terraformu a aplikovanie zmien vo vCenter.

Zmeny týchto parametrov vo virtuálnom zariadení vyžadujú jeho reštart. Pôvodne bolo aj v pláne vytvoriť prenastavenie virtuálnej sieťovej karty (**IP adresa!**, maska podsiete...) či prípadné pridanie novej virtuálnej sieťovej karty. To ale vyžaduje v Terraforme vymazať pôvodné virtuálne zariadenie a nahradiť ho novým, preto pridanie týchto parametrov v diplomovej práci nebolo realizované.

5.8 Vymazanie virtuálneho zariadenia

Poslednou Jenkins úlohou pre spravovanie virtuálnych zariadení vo vCenter je ich vymazanie. V Jenkins zadávacom formulári užívateľ iba vyberie, ktoré virtuálne zariadenie chce vymazať. Následne dôjde k vymazaniu konfigurácie cieľového virtuálneho zariadenia a po spustení Terraformu je zariadenie vymazané.

5.9 Diskusia riešenia

Celkovo bolo v rámci diplomovej práce vytvorených šesť Jenkins úloh, ktoré slúžia na jednoduché spravovanie virtuálnej infraštruktúry v Algotech dátovom centre. Testovanie Jenkins úloh prebiehalo na lokálnom počítači s operačným systémom Windows. Po niekoľkých úpravách zdrojových kódov pre Jenkins úlohy je možné výsledné riešenie nasadiť aj na zariadeniach s operačným systémom Linux. Pred nasadením nástroja je potrebné okrem inštalácií Jenkins, Terraform a Python ešte doinštalovať potrebné Jenkins pluginy (Credentials a Extended Choice Parameter) a nepatrne upraviť zdrojové kódy. Táto úprava iba zahŕňa zmenu cesty k pracovnému adresáru. V rámci výsledného riešenia boli aj implementované dve dodatočné Jenkins úlohy na predpripravenie pracovných adresárov o potrebné konfiguračné súbory.

Jenkins administrátor môže vytvoriť dodatočných užívateľov (napr. sieťového administrátora), ktorí budú mať oprávnenie pristupovať iba k určitej sade Jenkins úloh (napr. iba spravovanie sietí v NSX-T).

Vytvorené riešenie možno dotatočne upravovať a rozširovať. Ďalšími možnými rozšíreniami sú spravovanie aj inej infraštruktúry (Tier-0 a Tier-1 smerovače) a služieb (DHCP, NAT) v NSX-T. Spravovanie buď centrálneho alebo lokálneho DHCP serveru pre každý segment by umožnilo odstrániť statické nastavenie IP adresy na virtuálnom zariadení. Podobne vytvorením Jenkins úlohy na vytváranie NAT pravidiel by bolo možné zabezpečiť pripojenie virtuálneho zariadenia do internetu.

V rámci vCenter, ako už bolo zmienené, by bolo možné vytvoriť Jenkins úlohy na autoinštaláciu operačného systému. Tým by sa odstránila nutnosť klonovať nové virtuálne zariadenia z vopred definovanej šablóny.

Zadávací formulár v Jenkins nie je interaktívny a nemožno v ňom napr. skryť alebo zobraziť voliteľné parametre. Ďalším možným námetom pre vylepšenie aktuálneho riešenia je vytvorenie iného užívateľského rozhrania. To by fungovalo na rovnakom princípe ako doterajšie užívateľské rozhranie v Jenkins. Zadané parametre z externého rozhrania by sa napr. zapísali do súboru. Jenkins by následne detekoval túto zmenu v nastavení a spustil by automaticky niektorú z vytvorených Jenkins úloh na úpravu virtuálnej infraštruktúry.

Hlavný cieľ diplomovej práce, a to automatizovať procesy od zadania vstupných parametrov až po vykonanie konfiguračných zmien buď v prostredí NSX-T alebo vCenter bol splnený. Vytvorené riešenie pre automatizáciu prenosu konfigurácie do vCenter a NSX-T primárne používa nástroje Jenkins a Terraform. V rámci výsledného riešenia bolo využité Jenkins užívateľské rozhranie, kde operátori Service desku môžu zadávať požadované nastavenia virtuálnych instancií v prostrediach NSX-T alebo vCenter.

5.9.1 Bezpečnosť vytvoreného riešenia

Implementované zdrojové kódy nie je nutné uchovávať v tajnosti, keďže neobsahujú citlivé údaje ako napr. užívateľské mená a heslá pre autentizáciu s API cieľového prostredia. Prihlasovacie údaje sú bezpečne uložené v Jenkins prostredníctvom Credentials pluginu. Vďaka tomuto pluginu Jenkins neodosiela prihlasovacie údaje v otvorenom texte. Dodatočné textové súbory, ktoré využíva Jenkins, neobsahujú citlivé údaje ako napr. IP adresy virtuálnych zariadení či Tier-1 smerovačov. Tieto súbory obsahujú iba názvy segmentov či iných entít.

Zraniteľný bod vo výslednom riešení ale predstavujú Terraform konfiguračné súbory. Tie môže neoprávnený užívateľ upravovať a meniť tým výsledné nastavenie cieľového prostredia. Tu je nutné ešte dodatočne zaistiť bezpečnostné princípy dôvernosti a integrity. V prípade straty konfiguračného súboru hrozí odstránenie aktuálnej konfigurácie cieľového prostredia. Ako možné riešenie sa ponúka vytvárať kópiu Terraform konfiguračného súboru na externú lokalitu. V prípade straty hlavného súboru by tak bol použitý záložný konfiguračný súbor.

Kapitola 6

Záver

SDN v spojení s technológiami ako NFV a virtualizácia tvoria dôležitú súčasť moderných dátových centier. Vďaka nim je možné centralizovať riadenie sietí, vytvárať virtuálne siete či efektívne využívať výpočtové zdroje hardvéru v dátových centrách. V teoretickej časti diplomovej práce bol predstavený pojem SDN a popísaná architektúra SDN ako aj výhody SDN oproti tradičným sieťam.

Práca sa ďalej venovala popisu používaných virtualizačných nástrojov vSphere a NSX-T v dátovom centre Algotechu. Ku obom nástrojom bol uvedený stručný úvod, pričom u nástroja NSX-T boli hlbšie rozobrané niektoré témy ako napr. logické prepínanie či smerovanie.

Oba zmienené nástroje možno ovládať buď prostredníctvom užívateľského rozhrania alebo API. Práve s využitím API a externých nástrojov, ako Jenkins a Terraform, bolo vytvorené riešenie, ktoré umožňuje automatizovať proces správy vybraných virtuálnych instancií (segmentov a virtuálnych zariadení) v prostrediach NSX-T a vCenter. V rámci výsledného riešenia praktického zadania diplomovej práce bolo tiež vytvorené v Jenkins jednoduché užívateľské rozhranie, pomocou ktorého možno externe ovládať zmieňované cieľové prostredia. Toto užívateľske rozhranie môže byť využité, napr. zamestnancami Algotech Service desku pre spravovanie časti virtuálnej infraštruktúry v dátovom centre. Prínos tejto diplomovej práce teda spočíva vo vytvorení nástroja pre správu virtuálnych instancií. Vo výslednom riešení je prenos požadovanej konfigurácie do cieľového prostredia plne automatizovaný. V práci bola tiež v krátkosti zhrnutá bezpečnosť výsledného riešenia, ako aj možné útoky na SDN.

Literatúra

[1] W. Odom, *CCNA 200-301 Official Cert Guide, Volume 2*, 22.12. 2019, ISBN: 978-1-58714-713-5

[2] Avianet: 10 Up and Coming Trend About Network Technology, [online], [cit. 10.3. 2023]. Dostupné: `https://www.avianet.aero/network-technology/`

[3] IBM: What is DevOps?, [online], [cit. 10.3. 2023]. Dostupné: `https://www.ibm.com/topics/devops`

[4] J. Edwards, *Getting Started with Intent-Based Networking*, [online], 23.2. 2021, [cit. 10.3. 2023]. Dostupné: `https://www.ibm.com/topics/devops`

[5] VMWare: What is Software-Defined Networking (SDN)?, [online], [cit. 10.3. 2023]. Dostupné: `https://www.vmware.com/topics/glossary/content/software-defined-networking.html`

[6] M. Awais, M. Asif, M. Bin Ahmad, T. Mahmood, S. Munir, *Comparative Analysis of Traditional and Software Defined Networks*, [online], 2021, [cit. 1.3. 2023]. Dostupné: `https://ieeexplore.ieee.org/document/9526236/citations?tabFilter=papers#citations`

[7] M. MIKÉSKA, *Správa a řízení datových sítí pomocí softwarově řízené sítě*, [online], 2019, [cit. 1.3. 2023]. Dostupné: `https://dspace.cvut.cz/handle/10467/83339`

[8] F. Holík, S. Neradová, *Softwarově definované sítě*, 2019, [online], [cit. 5.3. 2023], ISBN: 978-80-7560-235-0 (pdf)

[9] D. Kreutz, F. M. V. Ramos, P. E. Veríssimo, C. E. Rothenberg, S. Azodolmolky, S. Uhlig, *Software-Defined Networking: A Comprehensive Survey*, in Proceedings of the IEEE, vol. 103, no. 1, pp. 14-76, [online], Jan. 2015, doi: 10.1109/JPROC.2014.2371999, [cit. 11.3. 2023]. Dostupné: `https://ieeexplore.ieee.org/abstract/document/6994333`

[10] RedHat: What is NFV?, [online], 2019, [cit. 10.9. 2023]. Dostupné: https://www.redhat.com/en/topics/virtualization/what-is-nfv

[11] K. Rangan, *A complete guide to software-defined networking*, 2021, [online], [cit. 10.4. 2023]. Dostupné: https://www.g2.com/articles/software-defined-networking

[12] ResearchGate: The basic SDN architecture, [online], [cit. 10.5. 2023]. Dostupné: https://www.researchgate.net/figure/The-basic-SDN-architecture_fig1_349468730

[13] S., Ahmad, A., H., Mir, *SDN Interfaces: Protocols, taxonomy and challenges*, 8.4. 2022, [online], [cit. 11.4. 2023]. Dostupné: https://www.mecs-press.org/ijwmt/ijwmt-v12-n2/IJWMT-V12-N2-2.pdf

[14] O. Blial, B. M. Mamoun, R. Benanini, *An Overview on SDN Architectures with Multiple Controllers*, Journal of computer networks and communications, 27.4. 2016, [online], [cit. 16.4. 2023]. Dostupné: https://www.hindawi.com/journals/jcnc/2016/9396525/

[15] G., Yangyang, *What Is OpenFlow?*, 5.8. 2021, [online], [cit. 20.4. 2023]. Dostupné: https://support.huawei.com/enterprise/en/doc/EDOC1100196737

[16] Open Networking Foundation, *OpenFlow Switch Specification version 1.5.1.*, 2015, [online], [cit. 20.4. 2023]. Dostupné: https://opennetworking.org/wp-content/uploads/2014/10/openflow-switch-v1.5.1.pdf

[17] Javatpoint: Software Defined Networking (SDN): Benefits and Challenges of Network Virtualization, [online], [cit. 15.3. 2023]. Dostupné: https://www.ibm.com/topics/sdn

[18] IBM: What Is Software-Defined Networking?, [online], [cit. 15.3. 2023]. Dostupné: https://www.ibm.com/topics/sdn

[19] Techpedia: Siete budúcnosti - SDN a NFV, [online], [cit. 15.4. 2023]. Dostupné: https://techpedia.fel.cvut.cz/browse/?fileId=343&objectId=65&currPage=1010

[20] M. Jehlička, *Laboratorní platforma softwarově definovaných datových sítí*, 2019, [online], [cit. 15.4. 2023]. Dostupné: https://dspace.cvut.cz/handle/10467/83346

[21] M. F. Akbaş, E. Karaarslan, C. Güngör, *A Preliminary Survey on the Security of Software-Defined Networks*, 2016, [online], [cit. 18.9. 2023]. Dostupné: https://www.researchgate.net/publication/320226086_A_Preliminary_Survey_on_the_Security_of_Software-Defined_Networks

[22] M. Rahouti, K. Xiong, Y. Xin, S. K. Jagatheesaperumal, M. Ayyash and M. Shaheed, "SDN Security Review: Threat Taxonomy, Implications, and Open Challenges,ïn IEEE Access, vol. 10, pp. 45820-45854, [online], 2022, doi: 10.1109/ACCESS.2022.3168972, [cit. 18.9. 2023]. Dostupné: https://ieeexplore.ieee.org/document/9760465

[23] T. A. V. Sattolo, S. Macwan, M. J. Vezina, A. Matrawy, *Classifying Poisoning Attacks in Software Defined Networking*, 2019, IEEE International Conference on Wireless for Space and Extreme Environments (WiSEE), Ottawa, ON, Canada, 2019, pp. 59-64, doi: 10.1109/WiSEE.2019.8920310, [online], [cit. 10.9. 2023]. Dostupné: https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=8920310

[24] Cisco: What Is SD-WAN?, [online], [cit. 22.9. 2023]. Dostupné: https://www.cisco.com/c/en/us/solutions/enterprise-networks/sd-wan/what-is-sd-wan.html

[25] F. Flachs, *Automatizovaný systém konfigurace SDN síťových zařízeních*, 2020, [online], [cit. 22.9. 2023]. Dostupné: https://dspace.cvut.cz/handle/10467/88059

[26] R2 Unified Technologies: What Is Software-Defined Access (SD Access)? [online], [cit. 23.9. 2023]. Dostupné: https://blog.r2ut.com/what-is-software-defined-access-sd-access

[27] FS Community: Complete Guide to Software-Defined Data Center, 2022, [online], [cit. 26.9. 2023]. Dostupné: https://community.fs.com/article/complete-guide-to-software-defined-data-center.html

[28] IBM: Software-Defined Data Centers, [online], [cit. 20.11. 2023] Dostupné: https://www.ibm.com/topics/software-defined-data-center

[29] VMware: VMware vSphere, the Industry-Leading Virtualization Platform, [online], [cit. 21.11. 2023]. Dostupné: https://www.vmware.com/content/dam/digitalmarketing/vmware/en/pdf/products/vsphere/vmware-vsphere-datasheet.pdf

[30] Onekobo: VMware vSphere Components and Features, [online], [cit. 22.11. 2023]. Dostupné: http://www.onekobo.com/Articles/vSphere/Obj00/Obj0-2.html

[31] B. Lee, *Everything in VMware vCenter Server 7 – New Features in vSphere 7*, 2020, [online], [cit. 23.11. 2023] Dostupné: https://www.altaro.com/vmware/vcenter-server-7-new-features/

[32] S. Varshney, *What Is NSX-T VMware*, 2023, [online], [cit. 10.12. 2023]. Dostupné: https://www.c-sharpcorner.com/article/what-is-nsx-t-vmware/

[33] S. Lahiji, *Introduction to VMware NSX*, 2020,[online], [cit. 10.12. 2023]. Dostupné: https://www.velements.net/2020/01/07/introduction-to-vmware-nsx/

[34] D. Abhijeet, *NSX-T 3 Components*, 2021, [online], [cit. 13.12. 2023]. Dostupné: https://blogs.virtualmaestro.in/2021/09/10/nsx-t-3-0-components/

[35] VMware: VMware NSX-T ® Reference Design Guide, [online], [cit. 14.12. 2023]. Dostupné: https://arabitnetwork.files.wordpress.com/2018/12/vmware-nsx-t-reference-design-guide.pdf

[36] Palo Alto Networks: Securing Applications in VMware NSX - Design Guide, [online], [cit. 18.12. 2023]. Dostupné: https://www.paloaltonetworks.com/apps/pan/public/downloadResource?pagePath=/content/pan/en_US/resources/guides/securing-applications-deployed-in-a-vmware-nsx-t-data-center-design-guide

[37] T. Nguyen, *NSX Reference Design Guide*, [2020], [online], [cit. 15.12. 2023]. Dostupné: https://nsx.techzone.vmware.com/resource/nsx-reference-design-guide#a-23-nsx-consumption-model

[38] Nuwan: NSX-T, Logical Switching – Overlay and vLAN-Backed Segments, [online], [cit. 20.12. 2023]. Dostupné: https://nuwan.vip/nsx-t-logical-switching/

[39] VMware: NSX Security Reference Design Guide, 2021, [online], [cit. 19.12.2024]. Dostupné: https://nsx.techzone.vmware.com/resource/nsx-security-reference-design-guide

[40] Phoneix NAP: What is Jenkins?, 2022, [online], [cit. 28.12. 2023]. Dostupné: https://phoenixnap.com/kb/what-is-jenkins

[41] HashiCorp: What is Terraform?, [online], [cit. 29.12. 2023]. Dostupné: https://developer.hashicorp.com/terraform/intro

Dodatok A

Zoznam použitých skratiek

AAA	Authentication, authorization, and accounting
ACL	Access Control List
API	Application Programming Interface
ARP	Address Resolution Protocol
ASIC	Application-specific Integrated Circuit
BGP	Border Gateway Protocol
CAM	Content-addressable Memory
CCP	Central Control Plane
CLI	Command Line Interface
CPU	Central Processing Unit
DDoS	Distributed Denial of Service
DevOps	Development Operations
DFW	Distributed Firewall
DHCP	Dynamic Host Configuration Protocol
DoS	Denial of Service
DR	Distributed Router
DRS	Distributed Resource Scheduler

FIB	Forwarding Information Base
FIS	Flow Instruction Set
ForCES	Forwarding and Control Element Separation
FPGA	Field Programmable Gate Array
GENEVE	Generic Network Virtualization Encapsulation
GFW	Gateway Firewall
GRE	Generic Routing Encapsulation
GUI	Graphical User Interface
HDD	Hard Disk Drive
HTTP	Hypertext Transfer Protocol
IaaS	Infrastructure as a Service
IBN	Intent-based Networking
ID	Identifier
IoT	Internet of Things
IP	Internet Protocol
IPv4	Internet Protocol version 4
IPv6	Internet Protocol version 6
IT	Information Technology
JSON	JavaScript Object Notation
LCP	Local Control Plane
LLDP	Link Layer Discovery Protocol
L2	Layer 2
L3	Layer 3
L4	Layer 4

MAC	Media Access Control
MPA	Management Plane Agent
MPLS	Multiprotocol Label Switching
NAT	Network Address Translation
NDP	Neighbor Discovery Protocol
NFV	Network Function Virtualization
NOS	Network Operating System
N-VDS	NSX-T Virtual Distributed Switch
ONF	Open Network Foundation
OSPF	Open Shortest Path First
OVSDB	Open vSwitch Database
OSI	Open Systems Interconnection
PaaS	Platform as a Service
PDU	Protocol Data Unit
POF	Protocol Oblivious Forwarding
QoS	Quality of Service
RAM	Random-access Memory
REST	Representational State Transfer
RIB	Routing Information Base
SaaS	Software as a Service
SD	Software-defined
SDDC	Software-defined Data Center
SDN	Software-defined Networking
SD-WAN	Software-defined Wide Area Network

SNMP	Simple Network Management Protocol
SR	Service Router
SSH	Secure Shell
SSL	Secure Sockets Layer
STP	Spanning Tree Protocol
TCAM	Ternary Content Addressable Memory
TCP	Transmission Control Protocol
TEP	Tunnel End Point
TLS	Transport Layer Security
TTL	Time to live
UDP	User Datagram Protocol
vDS	vSphere Distributed Switch
VLAN	Virtual Local Area Network
VM	Virtual Machine
VNI	Virtual Network Identifier
vNIC	virtual Network Interface Card
VPN	Virtual Private Network
VxLAN	Virtual extensible Local Area Network
WAN	Wide Area Network
5G	Fifth-Generation

Dodatok B

Zoznam priložených súborov

```
algotech/
├── nsxt/
│   ├── jenkins_job_Create_Segment.txt
│   ├── jenkins_job_delete_segment.txt
│   ├── jenkins_job_update_segment.txt
│   ├── nsx_api_and_prepare_workspace.py
│   ├── nsx_param_check.py
│   ├── prepare_working_directory.txt
│   ├── terraform_create_segment.py
│   ├── terraform_delete_segment.py
│   ├── terraform_update_segment.py
│   └── text_editing.py
└── vcenter/
    ├── create_VM.py
    ├── delete_VM.py
    ├── jenkins_job_create_VM.txt
    ├── jenkins_job_delete_VM.txt
    ├── jenkins_job_prepare_vcenter_working
    ├── jenkins_job_update_VM.txt
    ├── nsx_api_and_prepare_workspace.py
    ├── text_editing.py
    ├── update_VM.py
    ├── vcenter_api_and_create_work_space.py
    └── vcenter_parameters_check.py
```