VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

Fakulta elektrotechniky
a komunikačních technologií

DIPLOMOVÁ PRÁCE

Brno, 2018 Bc. Klára Moravcová

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ
FACULTY OF ELECTRICAL ENGINEERING AND COMMUNICATION

ÚSTAV TELEKOMUNIKACÍ
DEPARTMENT OF TELECOMMUNICATIONS

TVORBA VIRTUÁLNÍCH SÍŤOVÝCH TOPOLOGIÍ POMOCÍ SOFTWAROVĚ DEFINOVANÝCH SÍTÍ
VIRTUAL NETWORK TOPOLOGY DESIGN BASED ON SOFTWARE DEFINED NETWORKS

DIPLOMOVÁ PRÁCE
MASTER'S THESIS

AUTOR PRÁCE
AUTHOR
Bc. Klára Moravcová

VEDOUCÍ PRÁCE
SUPERVISOR
doc. Ing. Vít Novotný, Ph.D.

BRNO 2018

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ

Diplomová práce

magisterský navazující studijní obor **Telekomunikační a informační technika**
Ústav telekomunikací

Studentka: Bc. Klára Moravcová ***ID:*** 164610
Ročník: 2 ***Akademický rok:*** 2017/18

NÁZEV TÉMATU:

Tvorba virtuálních síťových topologií pomocí softwarově definovaných sítí

POKYNY PRO VYPRACOVÁNÍ:

Seznamte se s problematikou virtualizace v oblasti informačních technologií a datových sítí. Podrobně se zaměřte na principy softwarově definovaných sítí (SDN), proberte možnosti, existující technologie a zhodnoťte výhody i slabiny tohoto řešení datových sítí. V závislosti na dostupnosti hardwarových a softwarových komponentů s podporou SDN navrhněte laboratorní úlohu pro předmět Architektura sítí, vytvořte pracoviště, úlohu zprovozněte a vytvořte k ní návod.

DOPORUČENÁ LITERATURA:

[1] STALLING, W. “Foundations of Modern Networking: SDN, NFV, QoE, IoT, and Cloud“

Addison-Wesley Professional, ISBN: 9780134175478, 10/2015

[2] BOMBAL, D. „Software Defined Networking (SDN)“, Infinite Skills, ISBN: 9781491976609, 02/2017

Termín zadání: 5.2.2018 ***Termín odevzdání:*** 21.5.2018

Vedoucí práce: doc. Ing. Vít Novotný, Ph.D.
Konzultant:

prof. Ing. Jiří Mišurec, CSc.
předseda oborové rady

ABSTRAKT

Tato práce se zabývá virtualizací datových sítí a to především konceptem softwarově definovaných sítí. V rámci SDN je popsána architektura, bezpečnostní rizika a rozdíly oproti tradičním sítím. Je zde popsán i protokol OpenFlow, který je nedílnou součástí konceptu SDN. V krátkosti je také zmíněn koncept NFV.
Cílem této práce je nalézt dostupná řešení, navrhnout laboratorní úlohu a zpracovat laboratorní návod. Byly vypracovány tabulky na trhu dostupných kontrolérů a přepínačů pro SDN. Návrh laboratorní úlohy byl uskutečněn s emulátorem Mininet a kontrolérem OpenDaylight. Samotná laboratorní úloha má za cíl seznámit studenty předmětu Architektura sítí s konceptem SDN a jeho reálnou implementací.

KLÍČOVÁ SLOVA

Mininet, OpenFlow, OpenDaylight, SDN, softwarově definované sítě

ABSTRACT

This Diploma thesis deals with virtualization of data networks, but mainly with concept of software-defined networking. Architecture, security risks and differences besides traditional networks are described within SDN framework. Description includes protocol OpenFlow, which is integral part of SDN concept. Brief summary mentions also NFV concept.
The goal of this thesis is to determine available solutions, invent laboratory task and compile laboratory manual. Tables with currently available controllers and commutators for SDN were created and pattern of laboratoral task was realized within Mininet emulator and OpenDaylight controller. Laboratoral task itself is designed with aim to apprize students of subject of Network architecture with SDN concept and its real implementation.

KEYWORDS

Mininet, OpenFlow, OpenDaylight, SDN, software defined networikng

MORAVCOVÁ, Klára. *Tvorba virtuálních síťových topologií pomocí softwarově definovaných sítí*. Brno, 2018, 81 s. Diplomová práce. Vysoké učení technické v Brně, Fakulta elektrotechniky a komunikačních technologií, Ústav telekomunikací. Vedoucí práce: doc. Ing. Vít Novotný, Ph.D.

PROHLÁŠENÍ

Prohlašuji, že svou diplomovou práci na téma „Tvorba virtuálních síťových topologií pomocí softwarově definovaných sítí“ jsem vypracoval(a) samostatně pod vedením vedoucího diplomové práce a s použitím odborné literatury a dalších informačních zdrojů, které jsou všechny citovány v práci a uvedeny v seznamu literatury na konci práce.

Jako autor(ka) uvedené diplomové práce dále prohlašuji, že v souvislosti s vytvořením této diplomové práce jsem neporušil(a) autorská práva třetích osob, zejména jsem nezasáhl(a) nedovoleným způsobem do cizích autorských práv osobnostních a/nebo majetkových a jsem si plně vědom(a) následků porušení ustanovení § 11 a následujících autorského zákona č. 121/2000 Sb., o právu autorském, o právech souvisejících s právem autorským a o změně některých zákonů (autorský zákon), ve znění pozdějších předpisů, včetně možných trestněprávních důsledků vyplývajících z ustanovení části druhé, hlavy VI. díl 4 Trestního zákoníku č. 40/2009 Sb.

Brno ....

podpis autora(-ky)

PODĚKOVÁNÍ

Ráda bych poděkovala vedoucímu diplomové práce panu doc. Ing. Vítu Novotnému, Ph.D. za odborné vedení, konzultace, trpělivost a podnětné návrhy k práci.

Brno ....

podpis autora(-ky)

Faculty of Electrical Engineering
and Communication
Brno University of Technology
Purkynova 118, CZ-61200 Brno
Czech Republic

http://www.six.feec.vutbr.cz

PODĚKOVÁNÍ

Výzkum popsaný v této diplomové práci byl realizován v laboratořích podpořených z projektu SIX; registrační číslo CZ.1.05/2.1.00/03.0072, operační program Výzkum a vývoj pro inovace.

Brno ....

podpis autora(-ky)

EVROPSKÁ UNIE
EVROPSKÝ FOND PRO REGIONÁLNÍ ROZVOJ
INVESTICE DO VAŠÍ BUDOUCNOSTI

OBSAH

Úvod **13**

1 Virtualizace datových sítí **15**

2 Softwarově definované sítě **16**
2.1 Architektura tradičních sítí 16
2.1.1 Struktura síťových zařízení 16
2.1.2 Komunikace v síti 17
2.1.3 Limity tradičních sítí 18
2.2 Architektura softwarově definovaných sítí 19
2.2.1 Komponenty 20
2.2.2 Komunikace v SDN sítích 23
2.3 Bezpečnostní rizika 23
2.4 Výhody a limity 24
2.4.1 Řešení SDN s více kontroléry 25

3 Architektura OpenFlow **26**
3.1 OpenFlow přepínač 26
3.1.1 Porty 26
3.1.2 OpenFlow tabulky 27
3.2 OpenFlow protokol 29
3.2.1 Zprávy 30
3.3 Verze 32

4 Virtualizace síťových funkcí **33**
4.1 SDN vs. NFV 33

5 Dostupná SDN řešení **34**
5.1 Kontroléry 34
5.1.1 Open-source kontroléry 34
5.1.2 Komerční kontroléry 34
5.2 Přepínače 35
5.2.1 Softwarové SDN přepínače 36
5.2.2 Hardwarové SDN přepínače 36

6 Nástroje pro tvorbu laboratorní úlohy **38**
6.1 Virtuální prostředí 38
6.1.1 Oracle VM VirtualBox 38

6.1.2 Ubuntu . . . 38
6.2 Wireshark . . . 39
6.3 Mininet . . . 40
6.3.1 Mininet CLI . . . 40
6.3.2 Mininet API . . . 41
6.3.3 Tvorba topologií . . . 43
6.4 OpenDaylight . . . 44
6.4.1 Ovládání . . . 44
6.4.2 OpenFlow Manager . . . 47

7 Popis laboratorní úlohy **49**
7.1 Praktická úloha . . . 50

8 Závěr **52**

Literatura **53**

Seznam symbolů, veličin a zkratek **56**

Seznam příloh **57**

A Konfigurace softwarového vybavení laboratorní úlohy **58**
A.1 VirtualBox . . . 58
A.2 Ubuntu . . . 58
A.3 Mininet . . . 59
A.4 OpenDaylight . . . 59
A.5 OpenFlow Manager (OFM) . . . 60

B Laboratorní návod **61**
B.1 Cíl laboratorní úlohy . . . 61
B.2 Vybavení pracoviště . . . 61
B.3 Teoretický úvod . . . 61
B.3.1 Architektura tradičních sítí . . . 61
B.3.2 Architektura softwarově definovaných sítí . . . 63
B.3.3 Architektura OpenFlow . . . 66
B.3.4 OpenFlow protokol . . . 68
B.4 Postup . . . 73
B.4.1 Úkol 1 . . . 73
B.4.2 Úkol 2 . . . 74
B.4.3 Úkol 3 . . . 76
B.4.4 Úkol 4 . . . 78

B.4.5 Úklid pracoviště 78
B.5 Seznam zkratek 79
B.6 Doporučená literatura 79

C Zdrojový kód **80**

D Obsah přiloženého CD **81**

SEZNAM OBRÁZKŮ

2.1 Struktura síťových zařízení v tradičních sítích 17
2.2 Architektura softwarově definovaných sítí 20
2.3 Komponenty SDN a komunikace mezi nimi [4] 21
3.1 Skladba OpenFlow přepínače [11] 27
3.2 Struktura záznamů v OpenFlow tabulce [6] 28
3.3 Příklad komunikace mezi přepínačem a kontrolérem 30
6.1 Grafické rozhraní programu Wireshark se zachycenými pakety [17]. . . 39
6.2 Architektura OpenDaylight kontroléru [20] 45
6.3 Konzola Karaf kontejneru a instalace funkcí 46
6.4 Grafické rozhraní OpenDaylight kontroléru 48
6.5 Grafické rozhraní aplikace OpenFlow Manager 48
7.1 Obecné schéma zapojení laboratorní úlohy 50
B.1 Struktura síťových zařízení v tradičních sítích 62
B.2 Architektura softwarově definovaných sítí 65
B.3 Komponenty SDN a komunikace mezi nimi 66
B.4 Skladba OpenFlow přepínače [11] 67
B.5 Struktura záznamů v OpenFlow tabulce [6] 68
B.6 Příklad komunikace mezi přepínačem a kontrolérem 69
B.7 Obecné schéma zapojení laboratorní úlohy 73
B.8 Úkol 1: Topologie vytvořena příkazem v mininet. 75
B.9 Úkol 3: Tabulka toků přepínače S2. 76

SEZNAM TABULEK

5.1 Open-source kontroléry 34
5.2 Komerční kontroléry 35
5.3 Softwarové přepínače 36
5.4 Hardwarové přepínače 36
6.1 Základní příkazy pro Mininet 41
6.2 Základní funkce kontroléru OpenDaylight 47
B.1 Základní příkazy pro Mininet 71

ÚVOD

Diplomová práce se zabývá popisem moderních síťových technologií s využitím virtualizace, kterým jsou například softwarově definované sítě a virtualizace síťových funkcí a dalšími tématy, která jsou s těmito koncepty úzce spojeny. Vlivem rychle se rozvíjejícího IT odvětví a tím zvyšujících se nároků na počítačové sítě bylo zapotřebí najít nové řešení síťové infrastruktury, neboť tradiční pojetí sítí, je již pro nasazování nových technologií nedostačující. Tyto sítě mají nedostatky při potřebě rychlého nasazování nových síťových služeb a aplikací a také je obtížné tyto sítě přizpůsobovat individuálním požadavkům organizací. Řízení sítě je decentralizované, kde každý síťový prvek má svou rozhodovací politiku a je svázán s implementovaným softwarem s určitými funkcemi bez možnosti změny. Naopak moderní síťové technologie využívají virtualizace, díky které lze měnit hardwarové prostředky dle aktuálních potřeb. Také oddělují řízení z každého síťového prvku a přesouvají ho do jednoho centrálního prvku, které poskytuje rozhraní pro implementaci dalších služeb a aplikací. Dalším znakem moderních sítí je otevřenost, kdy software již nebývá svázán s hardwarem, ale lze využít obecné zařízení se softwarem dle vlastního výběru.

V úvodu je popsán obecný koncept virtualizace a využití v datových sítích. V další kapitole je rozsáhle popsán koncept softwarově definovaných sítí, jehož základním principem je oddělení řídicí části a části transportu dat z hardwarového zařízení. Je zde rozebrána architektura, bezpečnost, výhody a limity tohoto konceptu. Pro lepší pochopení odlišností SDN a jeho výhod je v této kapitole také popsána architektura tradičních IP sítí a její limity. Nevýhody tradičních sítí dávají motivaci pro budování moderních konceptů, které tyto limity eliminují.

Ve třetí kapitole nalezneme architekturu OpenFlow, která je nedílnou součástí konceptu SDN. V OpenFlow architektuře je popsána podoba přepínače a protokol, který slouží pro komunikaci mezi kontrolérem a přepínačem. OpenFlow není jediným protokolem pro komunikaci mezi kontrolérem a přepínačem, ale je jediným standardem pro SDN a je široce využívaný, proto je v této práci rozebrán právě tento protokol.

V následující kapitole je také zmíněn koncept virtualizace síťových funkcí a jeho vztah k SDN. Pro tuto práci není zmíněný koncept stěžejní a v rámci tvorby laboratorní úlohy nebude uvažován. Dále jsou vypracovány souhrny aktuálně dostupných řešení kontrolérů a přepínačů s podporou technologie SDN. Na trhu je již dnes velké množství komerčních i open-source projektů a firem, které vytváří různá řešení pro použití v SDN. Vedle kontrolérů a přepínačů, které jsou v práci zmapovány, se jedná i o různé aplikace řešící mnohé aspekty sítí – například aplikace pro zvýšení bezpečnosti, monitoring, analýzy, směrování, apod.

Pro návrh laboratorní úlohy je podstatné softwarové vybavení. Je využit virtuální stroj bežící v Oracle VM Machine s distribucí linuxového operačního systému Ubuntu. Součástí návrhu laboratorní úlohy je i síťový emulátor Mininet, díky kterému lze vytvářet různé síťové topologie i při absenci hardwarových zařízení. Celou síť je tedy možné sestavit pouze na jednom virtuálním stroji. Dále je využit kontrolér OpenDaylight a síťový analyzátor Wireshark.

K laboratorní úloze je vytvořen návod, který se skládá z teoretického úvodu a několika praktických úkolů. Cílem této úlohy je seznámit studenty s konceptem SDN a poukázat na rozdíly oproti tradičním sítím. Praktická úloha poskytuje náhled do implementace tohoto konceptu.

1 VIRTUALIZACE DATOVÝCH SÍTÍ

Pojem virtualizace je obecně definován jako schopnost simulace hardwarové platformy do softwaru. Mohou to být zařízení jako servery, paměťové zařízení či síťové zdroje. Virtualizované funkce jsou odděleny od hardwaru a simulovány jako virtuální instance, které jsou schopny fungovat stejně jako tradiční hardwarové řešení. Tyto virtuální instance hostuje hardwarové zařízení, které již může být obecné bez specifického využití. Jedno hardwarové zařízení může být použito pro více virtuálních zařízení, které lze dle potřeby snadno zapínat či vypínat a také jim měnit hardwarové prostředky. Virtualizovaná řešení jsou ve výsledku typicky mnohem přínosnější, škálovatelnější a nákladově efektivnější, než tradiční řešení založené na hardwaru.

Virtualizace datových sítí je pojmem zahrnující možnosti vytvářet logické virtuální sítě, které jsou oddělené od síťového hardwaru tak, aby se zajistila lepší integrace sítě a podpora stále narůstajícího virtuálního prostředí. Abstrahuje se přípojení k síti a služby, které jsou tradičně dodávány prostřednictvím hardwaru. Tyto funkce běží nad fyzickou sítí v hypervisoru. Mimo služeb 2. a 3. vrstvy, jako je přepínání a směrování, síťová virtualizace zahrnuje i služby 4.-7. vrstvy, včetně vyrovnávání zátěže v síti a vyrovnávání zatížení serverů. Virtualizace sítí řeší řadu problémů v oblasti vytváření sítí v dnešních datových centrech. Pomáhá organizacím centrálně naprogramovat a zajišťovat sítě, aniž by se musely fyzicky přizpůsobovat základní fyzické a síťové struktury.

Díky virtualizaci mohou společnosti využívat efektivitu a agilitu výpočetních a úložných zdrojů založených na softwaru. V dnešní době jsou dva koncepty moderních sítí, které se zaměřují na virtualizaci datových sítí. Prvním je SDN (Software-Defined Networking) (viz 2) a druhým NFV (Network Functions Virtualization) (viz 4) [1].

2 SOFTWAROVĚ DEFINOVANÉ SÍTĚ

Oproti tradičním sítím, je SDN moderní architektura sítí, jejíž princip je fyzické oddělení datové a řídicí části. Díky tomuto oddělení se řízení sítě stalo plně programovatelným [2].

Snaha vytvořit více programovatelné sítě tu byla již od počátku 90. let, kdy výzkumníci chtěli vytvořit rozmanitější aplikace než jen doposud využívané pro přenos souborů a e-mailů pro vědce a také protokoly pro zlepšení síťové komunikace. Na počátku 21. století, vlivem zvyšujícího se objemu provozu a důrazu na spolehlivost, byla snaha najít řešení pro lepší přístupy k funkcím správy sítě. Směrovače a přepínače mají těsnou integraci řídicí části a části transportu dat, což představuje komplikace při konfiguraci a ladění chování těchto zařízení. Tehdy vznikla myšlenka rozdělit datové a řídicí části a vytvořit tak centralizované řízení sítě. Nicméně termín SDN byl poprvé použit v rámci OpenFlow projektu na Stanfordské univerzitě, který byl vyvíjen od roku 2007 [2].

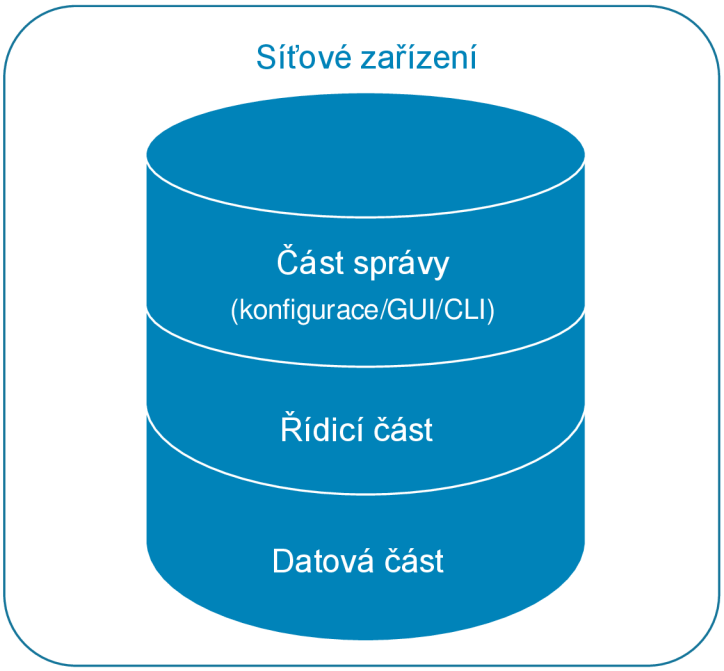
2.1 Architektura tradičních sítí

Pro lepší pochopení fungování softwarově definovaných sítí a rozdílů oproti dnes používaným sítím, je dále popsán základ architektury tradičních sítí.

2.1.1 Struktura síťových zařízení

Funkcionalita standardních síťových prvků, která je zobrazena na obr. 2.1, lze rozdělit do tří částí: část správy, řídicí část a část transportu dat [3]. Všechny tyto části jsou pevně implementovány v síťovém zařízení a jsou dále popsány:

- **Část transportu dat (data plane)** obsahuje porty a přepojovací pole pro příjem a odesílání paketů. Zajišťuje tedy přepojování provozu na základě rozhodnutí která stanovila řídicí část. Dále obsahuje vyrovnávací paměť paketů, plánování paketů a úpravu záhlaví [3],[4]. Pokud se však informace v záhlaví příchozích paketů nenachází ve směrovací tabulce nebo pokud této části nepřísluší zpracování daných paketů, je zpracování předáno řídicí části.
- **Řídicí část (control plane)** rozhoduje jak zacházet se síťovým provozem na základě definované politiky [3]. Udržuje aktuální informace ve směrovací tabulce, aby datová část mohla odbavit co nejvíce provozu. Řídicí část je také zodpovědná za zpracování informací od různých řídicích protokolů [4].
- **Část správy (management plane)** obsahuje možnosti konfigurace a monitoringu. Pomocí této vrstvy architektury definujeme síťovou politiku.



Obr. 2.1: Struktura síťových zařízení v tradičních sítích

2.1.2 Komunikace v síti

Další důležitou charakteristikou tradičních sítí je způsob předávání datových jednotek mezi zařízeními. Zde jsou popsány základní myšlenky předávání datových jednotek na 2. a 3. vrstvě (označení vrstev je dle ISO/OSI modelu):

Přepínání rámců na 2. vrstvě

Obecně přepínání na L2 může být podle identifikátoru cesty (u technologie ATM jsou to VCI/VPI) nebo na základě cílové adresy. Nejrozšířenější technologií v lokálních sítích je Ethernet. Komunikace u této technologie probíhá zasíláním rámců s využitím fyzické adresy (MAC adresy). Prvkem v síti již dnes bývá výhradně přepínač (L2 switch), který přepíná rámce na základě cílové MAC adresy uvedené v přijatém rámci. Přepínač obsahuje MAC tabulku, kde jsou uloženy MAC adresy cílových stanic a jejich cílové rozhraní (tedy informace o tom, na jaký port má být rámec s určitou cílovou adresou odeslán).

Záznamy do MAC tabulky si přepínač postupně přidává na základě procházejících rámců. U příchozího rámce si přečte cílovou adresu stanice a hledá shodu ve své tabulce. Pokud je shoda nalezena, zašle rámec na uvedený port. Pokud záznam pro tuto adresu nalezen není, zašle rámec na všechny porty mimo příchozího portu. MAC adresu a port cílové stanice si přepínač zaznamená při odpovědi. Také si přečte

zdrojovou MAC adresu stanice. Tuto adresu a port, ze kterého rámec přišel, si přidá do své tabulky. Takto se postupně učí MAC adresy a porty všech koncových stanic v síti.

Směrování paketů na 3. vrstvě

Komunikace na 3. vrstvě je označována jako směrování a využívá IP adresy. Prvek v síti, který provádí směrování je obvykle směrovač (router) nebo přepínač 3. vrstvy (L3 switch). Každý směrovač má směrovací tabulku, která obsahuje obraz topologie, podle které se rozhoduje kam daný paket zašle. Záznamy v tabulce mohou být statické (nastavené správcem sítě) nebo dynamické (vytvářené pomocí směrovacího protokolu). V obecné rovině funkce směrování obsahuje protokol pro sběr informací o topologii dané sítě, provozu, kapacitě linek a v případě dynamického směrování i algoritmus pro navrhování tras sítí.

Směrovací protokoly můžeme rozdělit na interní směrovací protokoly používané uvnitř autonomního systému – mezi takové protokoly patří například RIP (Routing Information Protocol), OSPF (Open Shortest Path First) a EIGRP (Enhanced Interior Gateway Routing Protocol) a externí směrovací protokoly používané pro směrování mezi autonomními systém – mezi takové protokoly patří BGP (Border Gateway Protocol).

Každý směrovač v síti si vytváří obraz o topologii, sbírá informace o konektivitě a zpoždění a dále vypočítává preferovanou trasu ke každé cílové adrese. Na základě těchto informací směruje datové jednotky sítí.

2.1.3 Limity tradičních sítí

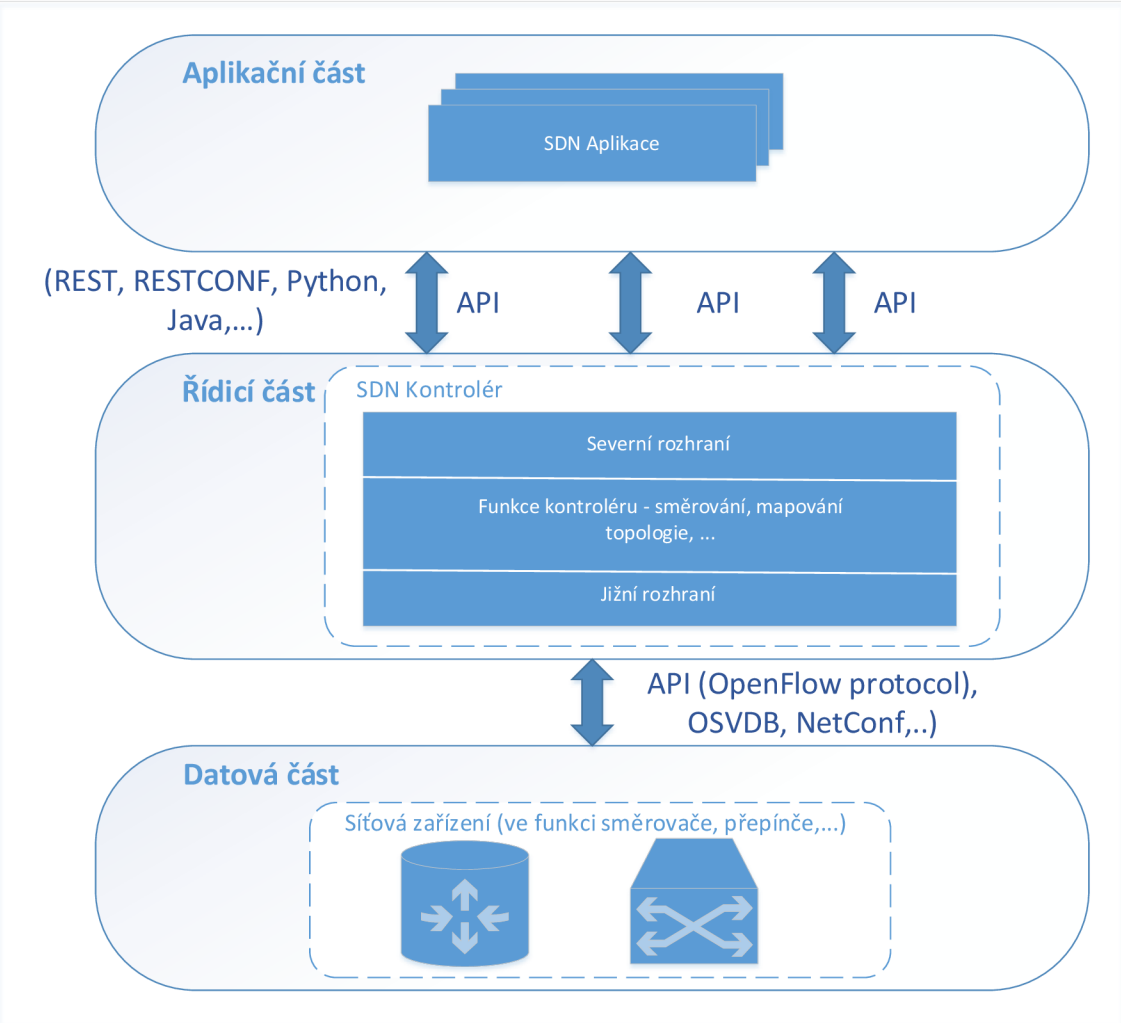
Takto hardwarově zaměřená infrastruktura má několik nedostatků, jako je obtížná správa, flexibilita, rozšiřitelnost a decentralizace. Kvůli svázanosti řídicí části a části přenosu dat je nutné konfigurovat každý síťový prvek zvlášť. Taková síť bývá náchylná na kolize a nevyváženosti zátěže. Hledání a odstraňování problémů je zde také obtížnější. Síťová zařízení obvykle podporují několik konfiguračních příkazů závislých na daném operačním systému či firmwaru, který je typicky proprietární. Síťoví administrátoři jsou tedy omezeni funkcemi, které výrobce implementoval do daného zařízení. Rozšíření je možné pouze aktualizací operačního systému či firmwaru, nebo zakoupení jiného síťového zařízení, které bude potřebné funkce zajišťovat. Obvykle se v jedné síti objevují síťová zařízení od různých výrobců s vlastním způsobem konfigurace, což způsobuje zvyšující se nároky na znalosti síťových administrátorů [2, 3, 5]. Tyto nedostatky dávají prostor pro vývoj moderních řešení, jako jsou právě SDN, které tyto limity odstraňují.

2.2 Architektura softwarově definovaných sítí

Jak již bylo uvedeno, SDN je architektura založena na fyzicky oddělené řídicí a datové části. Řídicí část již není uvnitř síťového zařízení, jako to je v tradiční architektuře, ale již na externím zařízení zvané kontrolér, které vzdáleně řídí jednu či více síťových jednotek jako jsou například přepínače a směrovače. Řízení celé sítě je tedy v této architektuře centralizované a díky SDN aplikacím, bežícím nad kontrolérem, je síť plně programovatelná [2].

Architektura SDN je zobrazena na obr. 2.2 a skládá se z následujících částí:

- V **datové části** pracují přepojovací zařízení, která přenášejí a zpracovávají data na základě rozhodnutí od řídicí části – tedy kontroléru. Zařízení tedy komunikuje s kontrolérem a je jím řízeno. Základní charakteristikou síťových zařízení pracující v této části jsou funkce jednoduchého přeposílání bez možnosti vlastního autonomního rozhodování[6]. Obsahuje tabulky a pravidla, na základě kterých se rozhoduje, jak bude naloženo s příchozími pakety s ohledem na informace jako jsou MAC adresy, IP adresy a VLAN ID. Zařízení může pakety přeposlat, zahodit, zpracovat či replikovat [4].
- **Jižní rozhraní** poskytuje logické propojení mezi SDN kontrolérem a síťovým prvkem operující v datové části (jako např. přepínač, virtuální přepínač, směrovač, firewall, atd.)[6]. Nejrozšířenějším API (Application Programming Interface) pro jižní rozhraní je protokol OpenFlow, který je popsán v kapitole 3. Mezi další API patří například OVSDB (Open vSwitch Database Management Protocol) a ForCES (Forwarding and Control Element Separation) [4].
- **Řídicí část** je v SDN sítích implementována jako jeden či více kontrolérů bežících na serverech či virtuálních serverech. Kontrolér sbírá směrovací informace od SDN přepínačů pro sestavení nejlepší cesty. Dále také přijímá a zpracovává aplikační události, zajišťuje bezpečnostní mechanismy, vytváří a udržuje informace o topologii přepínačů, sbírá informace o provozu procházející danou sítí a konfiguruje parametry a atributy přepínače [6].
- **Severní rozhraní** umožňuje aplikacím přístup k funkcím a službám řídicí části. Tak jako je pro jižní rozhraní definován OpenFlow standard, pro severní rozhraní žádný takový standard definován není. Pro severní rozhraní jsou použity různými výrobci různá řešení. Příkladem implementace severního rozhraní je REST API (Representational State Transfer), RESTful API, Python API či Java API [4][6].
- **Aplikační část** obsahuje aplikace a služby, které definují, upravují a řídí chování a prostředky dané sítě [6]. Díky těmto aplikacím, které mohou být aplikacemi třetích stran, je síť plně programovatelná a lze jimi rozšířit stávající funkce sítě.



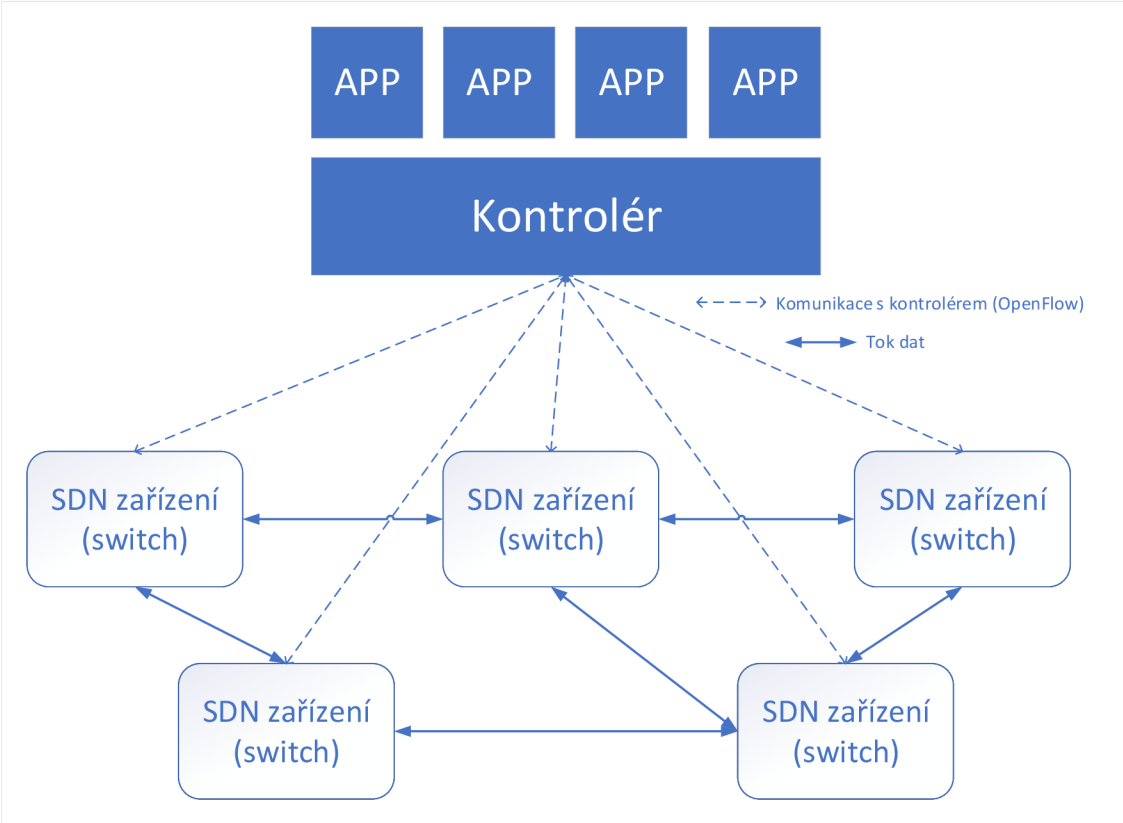
Obr. 2.2: Architektura softwarově definovaných sítí

2.2.1 Komponenty

Mezi základní komponenty, které jsou graficky popsány na obr. 2.3, patří SDN síťové zařízení, kontrolér a aplikace.

SDN síťové zařízení

Jak již bylo naznačeno v části 2.2, tato zařízení pracují v datové rovině, rozhodují jak naloží s příchozím paketem a jsou řízeny kontrolérem. Tato zařízení také obsahují instrukce, podle kterých se zařízení rozhoduje. Kontrolér definuje tok (flow), který je popsán jako sada paketů přenášených z jednoho koncového bodu na druhý. Stejná pravidla se použijí na všechny pakety patřící do jednoho toku [4]. Na základě takto aplikovaných pravidel se SDN přepínač může chovat jako směrovač, přepínač, firewall, NAT (Network Address Translator), nebo jako kombinace zmíněných [2].



Obr. 2.3: Komponenty SDN a komunikace mezi nimi [4]

Na zařízení je pak tabulka těchto toků (flow table) s jednotlivými záznamy seřazenými dle priority, kde u každého toku je také informace jak s příchozími pakety daného toku má být naloženo. Paket na základě této informace může být přeposlán na určitý port či na všechny porty nebo být zahozen.

S příchodem paketu je prohledána tabulka toků a pokud je nalezena shoda, aplikují se příslušná pravidla. Pokud shoda nalezena není, paket bude zahozen či předán kontroléru a to v závislosti na konfiguraci přepínače [4].

Kontrolér

Kontrolér udržuje přehled o celé síti, řídí všechna SDN zařízení v dané síti a poskytuje jižní rozhraní pro aplikace. Vkládáním záznamů do tabulek toků přepínačů implementujte pravidla týkající se směrování, přeposílání a vyvažování zátěže [4].

Jak je patrno z obr. 2.2, kontrolér se skládá z několika modulů – rozhraní pro severní a jižní API a jádro s několika aplikacemi bežící mezi těmito rozhraními. Mezi základní funkce jádra kontroléru patří [4]:

- **Objevování zařízení koncových uživatelů** jako jsou například notebooky, osobní počítače, tiskárny, mobilní zařízení, apod.

- **Objevování síťových zařízení**, která utvářejí infrastrukturu sítě jako například přepínače, směrovače a bezdrátové přístupové body.
- **Správa topologie síťových zařízení** udržuje podrobné informace o propojení síťových zařízení mezi sebou a koncovými uživateli, ke kterým jsou přímo připojeny.
- Modul **řízení toku** udržuje databázi toků, které jsou řízeny kontrolérem, a provádí synchronizaci položek databáze s SDN zařízeními.

Kontrolér vytváří záznamy pro veškerý provoz v síti. Existují dva režimy pro vkládání záznamů toků do přepínače – reaktivní a proaktivní.

- V **reaktivní režimu** kontrolér zpočátku neimplementuje žádná pravidla, která by určovala jak zacházet s příchozími rámci. Při přijetí rámce se přepínač dotáže kontroléru jak daný rámec zpracovat. Kontrolér na základě informací o topologii rozhodne, jak daný rámec zpracovat a vytvoří záznam v tabulce toků. Tento záznam platí pro všechny datové jednotky stejného typu. Přepínač se tedy dotazuje na každý rámec, u kterého nenašel shodu v tabulce toků – tím se tabulka postupně plní záznamy. Tento režim efektivně využívá paměť tabulek toků, ale generuje mnoho zpráv pro kontrolér a případě rozsáhlé sítě může vnést vyšší zpoždění, a proto je třeba mít kontrolér co nejblíže k přepínačům. Při výpadku kontroléru má síť pouze omezené možnosti v přenášení rámců [7].
- U **proaktivního režimu** kontrolér vytvoří záznamy v tabulkách toků na základě informací o topologii pro všechny možné toky ještě před zahájením provozu v síti. V tomto režimu se generuje méně zpráv pro kontrolér. Jelikož jsou toky předem definovány, nedochází k dalšímu zpoždění z důvodu nastavování toků a výpadek kontroléru nezpůsobí přerušení provozu v síti [7].

SDN aplikace

SDN aplikace běží nad kontrolérem a se sítí jsou propojeny přes severní rozhraní kontroléru, přes které může konfigurovat toky pro směrování paketů nejlepší cestou mezi dvěma koncovými body, vyvažovat provozní zatížení napříč několika cestami, reagovat na změny v síťové topologii (např. selhání spojení nebo přidání nových spojů a zařízení či přesměrování provozu pro účely kontroly, autentizace, segregace a jiných podobných bezpečnostních úkolů). Dále to také mohou být aplikace jako například firewall, monitorovací systém či GUI (Graphical User Interface) pro správu a konfiguraci kontroléru, přepínačů a aplikací [4].

2.2.2 Komunikace v SDN sítích

V SDN sítích jsou funkce přepojování centralizované uvnitř kontroléru, který má ucelené informace o stavu celé sítě. Díky těmto informacím může sestavit nejkratší cestu a může aplikovat směrovací politiku stanovenou od SDN aplikace. SDN přepínače fungující v datové části jsou oproštěny od zpracovatelské a úložné zátěže [6]. Centralizované směrovací aplikace provádí dvě funkce:

- **Vyhledávání spojení (link discovery)** – pro funkci směrování musí být známy vazby datových částí mezi směrovači v dané síti a také musí být známá spojení mezi směrovačem a hostitelským systémem [6].
- **Správce topologie (topology manager)** – udržuje informace o topologii sítě a vypočítává trasy v síti. Výpočet trasy je založen na určení nejkratší cesty mezi dvěma uzly nebo mezi uzlem a hostitelem [6].

2.3 Bezpečnostní rizika

S centralizovaným řízením a programovatelností sítě se SDN potýká s jistými bezpečnostními riziky. S centralizovaným řízením a omezenou velikostí tabulky toku se potenciálně zvyšují možnosti DoS (Denial-of-Service) [1] útoků a neautorizovaného přístupu [8]. Pokud útočník získá přístup ke kontroléru, získává tím kontrolu nad celou sítí. Může pak například měnit údaje v tabulkách toků (viz 3.1.2) v jeho prospěch [9].

SDN umožňuje integraci aplikací třetích stran do architektury, které rozšiřují stávající funkce a služby dané sítě. Nicméně jsou zde bezpečnostní rizika v důvěryhodnosti používaných aplikací. Aplikace může mít stejně škodlivý vliv jako napadený kontrolér. Může se jednat i o chyby v kódu aplikace, které vnesou do systému neúmyslné zranitelnosti. Tyto zranitelnosti však nemusí vnést do architektury pouze aplikace třetích stran, ale i samotný kontrolér či nedbalá konfigurace [9].

Další kategorií možné zranitelnosti spočívá v úniku dat. Přepínač má ve své tabulce toků definovány akce pro různé pakety. Jednou z akcí může být i předání zpracování určitých paketů kontroléru. Pokud útočník analýzou provozu zjistí, které z paketů jsou předány kontroléru, může vytvářet falešné požadavky pro kontrolér, ty mohou vést k DoS útoku. Dalším příkladem úniku dat je únik klíčů a certifikátů [9].

Zabezpečení sítě proti těmto hrozbám může probíhat na různých vrstvách architektury a to na aplikační, řídicí či datové různými způsoby. Například využívat

[1] Je útokem, který znefunkční či znepřístupní službu pro ostatní uživatele. Nejčastěji je tohoto efektu dosaženo přehlcením požadavků na dané zařízení. Zařízení má omezenou paměť pro zpracování a při velkém množství požadavků nestíhá všechny vyřídit a pak se služba stává nedostupnou.

zašifrované komunikace, kryptografické ověřování, podpůrné aplikace apod. Existuje několik řešení a výzkumů zabývající se zvýšením bezpečnosti. Obecně téma bezpečnosti je velmi rozsáhlé, nicméně není dále předmětem této práce.

2.4 Výhody a limity

Z porovnání SDN s tradičními sítěmi jsou již některé z výhod těchto moderních sítí zřejmé. Logická centralizace řídicí části nám poskytuje jednodušší modifikaci sítě, s menší náchylností na chyby, jednotným a centralizovaným způsobem přes vyšší programovací jazyky – oproti zařízením konvenčních sítí, které se konfigurují specifickými nízkoúrovňovými příkazy. Navíc je celé řízení konfigurováno v jednom místě a to v kontroléru. Tím se velice zjednodušují nároky na administrátora. Řídicí software běžící na kontroléru může automaticky reagovat na změny v síti a díky globální znalosti topologie a stavu sítě zjednodušuje vývoj specifických síťových služeb a aplikací. Další nespornou výhodou je nezávislost na výrobci – funkce zařízení již nejsou omezována implementací od výrobce a verzí daného firmwaru, ale je na administrátorovi, jaký software a jaké funkce bude ve své síti využívat [3]. Díky tomu lze rychleji přidávat a inovovat funkce bez nutnosti změny fyzických zařízení [13].

Nicméně, jsou zde i limity, které brání rychlejšímu nasazování SDN. Z pohledu vývoje se jedná o problém podrobnější spolupráce vývojářů s výrobci čipů a stávajícího hardwaru a nedostatečná dokumentace podpory OpenFlow od dodavatelů softwaru či hardwaru. Někteří výrobci hardwaru dokonce nepodporují standard v plném rozsahu tím, že jejich zařízení nemohou upravovat záhlaví paketu, což omezuje jejich užitečnost při směrování paketů nebo přepínání aplikací. Velké množství zařízení není schopno detekovat VLAN (Virtual Local Area Network) a MPLS (Multiprotocol Label Switching) značky nebo dokonce IPv6 [13]. Kvůli centralizovanému řízení sítě jsou kladeny vysoké nároky na kontroléry. Je nutné zajistit velký výpočetní výkon (závisle na velikosti sítě a počtu použitých aplikací) a vysokou dostupnost. Kontroléry také čelí problémům se škálovatelností[5].

Výše zmíněné limity lze díky programovatelnosti sítě alespoň částečně odstranit různými aplikacemi, které se zaměřují například zajištění konzistence pravidel mezi kontrolérem a přepínači či na rychlé obnovení kontroléru při výpadku. Je však možné vytvářet individuální řešení pro různá síťová prostředí, jako například pro bezdrátové, optické a domácí sítě. Dále také nástroje pro správu, bezpečnost nebo vyrovnávání zátěže [5].

2.4.1 Řešení SDN s více kontroléry

V dosavadním textu byl zmiňován a uvažován vždy pouze jeden kontrolér, což spolu s centralizovaným řízením přináší ve větších sítích řadu problémů. Především se jedná o škálovatelnost a vysokou dostupnost. U vysoké dostupnosti se jedná hlavně o dva aspekty – redundanci, která je potřebná v případě výpadku kontroléru a bezpečnost, kdy po neoprávněném získání přístupu ke kontroléru správce ztrácí kontrolu nad řízením celé sítě [10].

Z těchto důvodů bylo nutné vytvořit architekturu SDN podporující více kontrolérů v síti, které spolu budou spolupracovat, aby dosáhly požadované úrovně výkonu a škálovatelnosti. Ze strany přepínače a jižního rozhraní je podpora více kontrolérů zajištěna protokolem OpenFlow od verze 1.3 . Z pohledu kontrolérů dělíme architekturu na dva různé koncepty [10]:

- **Logicky centralizovaná** architektura využívá výhod konceptu více kontrolérů, ale zároveň z pohledu nižších vrstev se jedná pouze o jeden kontrolér. Provoz je pak distribuován mezi všechny kontroléry v síti. V této architektuře má každý kontrolér stejnou odpovědnost. Vždy vědí o každé změně v síti a okamžitě sdílejí stejné informace díky synchronizaci sítě.
  Příkladem implementace této architektury jsou například projekty: ONIX, HyperFlow, ONOS, DISCO či ELASTICON.
- V **logicky distribuované** architektuře jsou kontroléry fyzicky i logicky distribuovány. Navíc má každý kontrolér pod správou jen určitou část sítě, u které může provádět řízení a rozhodování.
  Příkladem implementace této architektury jsou například projekty: KANDOO či ORION.

3 ARCHITEKTURA OPENFLOW

OpenFlow je základním prvkem a jediným standardem pro vytváření řešení SDN. Definuje API mezi kontrolérem a přepínači, přes které je kontroléru umožněno měnit záznamy v tabulkách toků v přepínači. Komunikace probíhá přes zabezpečený kanál, který je běžně založen na TLS (Transport Layer Security) asymetrické kryptografii. Dále také určuje jak by zařízení měla reagovat v různých situacích a jak by měla reagovat na příkazy od kontroléru. Obsahuje sadu zpráv, které jsou odesílány v obou směrech [4].

V následujícím textu této kapitoly bylo čerpáno zejména ze zdroje [11].

3.1 OpenFlow přepínač

OpenFlow přepínač je zobrazen v obr. 3.1 a obsahuje jednu či více tabulek toků, tabulku skupin a tabulku měření. Dále obsahuje jeden či více OpenFlow kanálů propojující přepínač a kontrolér, přes který spolu komunikují pomocí OpenFlow protokolu. OpenFlow také definuje tři druhy portů: fyzické, logické a rezervované.

Funkcí tohoto přepínače je po přijetí paketu hledání shody v tabulce toků a na základě priority nalezené shody provést příslušnou akci. Paket může být poslán dál z lokálního portu s případnou úpravou některých polí v hlavičce, zahozen či předán kontroléru přes zabezpečený kanál. Pokud není nalezna žádná shoda, pak záleží na konfiguraci – paket může být například zahozen, poslán kontroléru či pokračovat na další tabulku toků. Veškerá pravidla a toky jsou nastavována kontrolérem [4, 11].

3.1.1 Porty

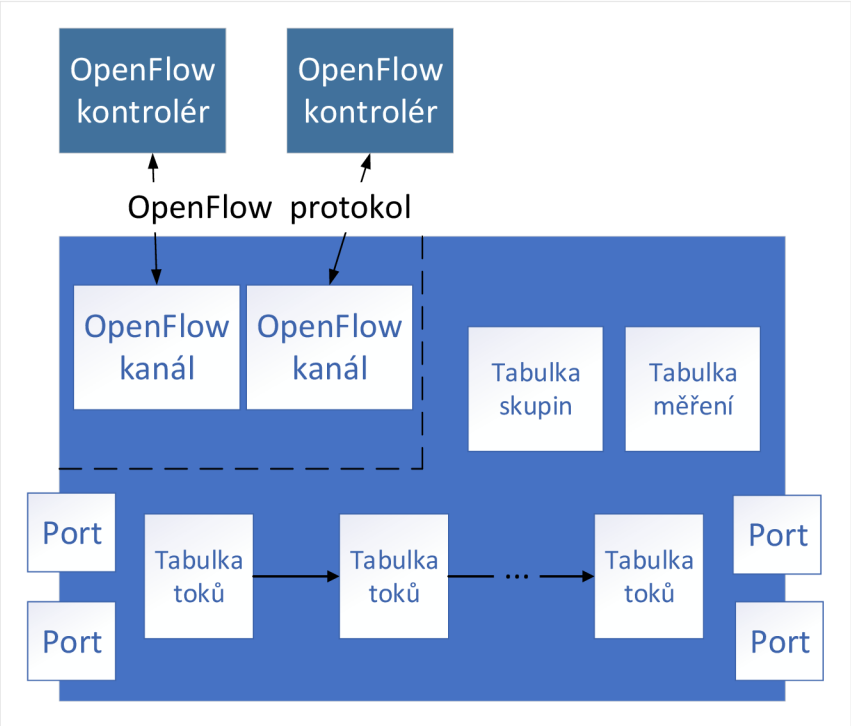
OpenFlow porty jsou síťová rozhraní pro předávání paketů mezi OpenFlow zpracováním a zbytku sítě. OpenFlow přepínače jsou přes tento port propojeny mezi sebou. Na hardwaru však nemusí být všechny porty povoleny pro OpenFlow. Přepínač musí podporovat fyzické, logické i rezervované porty.

Fyzické porty

Fyzické OpenFlow porty jsou definované přepínače a odpovídají hardwarovému rozhraní přepínače. Může se jednat například o ethernetové porty.

Logické porty

Logické porty nekorespondují s hardwarovým rozhraním a jsou abstrakcí vyšší úrovně. Mohou být definovány v přepínači pomocí metod, které nejsou definovány OpenFlow. Například agregace linek, tunely či loopbacky. Mohou zahrnovat zapouzdření



Obr. 3.1: Skladba OpenFlow přepínače [11]

paketů a mohou mapovat různé fyzické porty. Tyto porty však musí být transparentní pro OpenFlow a komunikovat s OpenFlow zpracováním, stejně jako fyzické porty.

Rezervované porty

Rezervované porty jsou definovány specifikací OpenFlow. Určují obecná přesměrovávání, například odeslání do kontroléru (port CONTROLLER), zaplavování nebo přesměrování pomocí metod, které nejsou definovány OpenFlow, jako je tradiční přepínání (port NORMAL).

3.1.2 OpenFlow tabulky

Tabulky toků

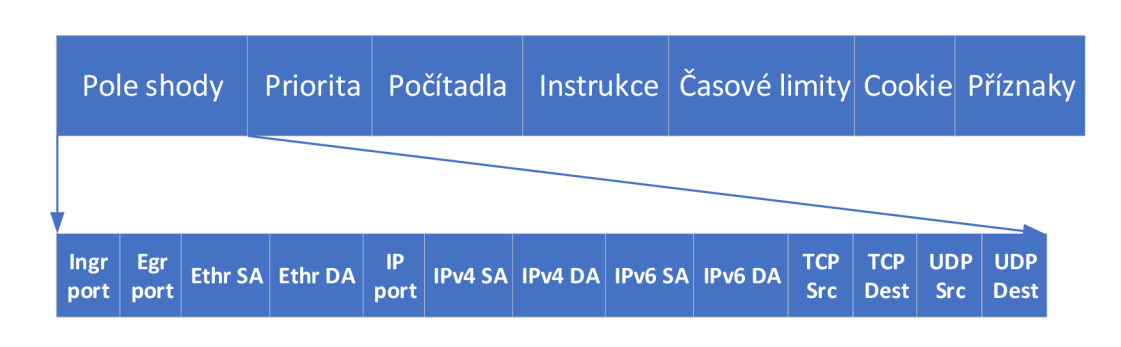
Každý paket, který přijde do přepínače, je porovnáván se záznamy v jedné či více tabulek toků. Každá tabulka obsahuje několik záznamů. Struktura záznamů je zobrazena na obr. 3.2.

Pokud je tabulek více, jsou zřetězeny. Tabulky jsou očíslovány, z nichž první tabulka začíná hodnotou 0. Při hledání shody se operuje nejprve s Tabulkou 0 a na základě výsledku mohou být využity i další tabulky. Při nalezení shody paketu se

záznamem v tabulce toků může být instrukcí tohoto záznamu odeslání paketu do jiné tabulky s vyšší hodnotou. Například může být záznam v Tabulce 0, který na základně určité IPv4 adresy odešle pakety dál ke zpracování do Tabulky 1, kde již bude shoda na základně protokolu transportní vrstvy a tudíž bude jiný záznam pro protokol TCP a jiný pro UDP (User Datagram Protocol).

Jednotlivá pole tabulky jsou popsána následovně:

- **Pole shody (match fields)** se využívá k výběru paketů, které se shodují v hodnotách tohoto pole. Shoda může být provedena, jak je vidět na obr. 3.2, na základě několika povinných hodnot. A to například na vstupním či výstupním portu přepínače, ethernetové adrese, verzi IP, IP adrese či TCP/UDP portech. Nicméně mohou být využívany i další pole, pokud jsou podporovány OpenFlow přepínačem. Mohou to být například VLAN ID, Tunnel ID či SCTP (Stream Control Transmission Protocol) port.
- Relativní **priorita (priority)** záznamu. Je to 16ti bitové pole, kde 0 reprezentuje nejnižší hodnotu.
- **Počítadla (counters)** se aktualizují, když je nalezena shoda s paketem.
- **Instrukce (instructions)**, které se mají provést pokud nastane shoda. Například předání na výstupní port či zahození.
- **Časové limity (timeoutes)** jsou definovány jako maximální doba před vypršením toku a způsobují odstranění záznamu z tabulky po určitém čase (hard_timeout) nebo po čase, kdy není nalezena shoda se žádným paketem (idle_timeout).
- **Cookie** je 64ti bitová hodnota určená kontrolérem. Může být použita kontrolérem k filtrování statistik toku, ke změně či smazání toku. Tato hodnota se nevyužívá při zpracování paketu.
- **Příznaky (flags)** mění způsob řízení záznamů tabulky.



Obr. 3.2: Struktura záznamů v OpenFlow tabulce [6]

Tabulka skupin

Tabulka skupin se skládá ze skupinových záznamů. Různé skupiny mohou reprezentovat různé abstrakce směrování, jako jsou vícesměrová a všesměrová vysílání. Každý záznam v této tabulce obsahuje:

- **Identifikátor skupiny (group identifier**, což je 32 bitové celé číslo, které identifikuje skupinu. Skupina je definovaná jako položka ve skupinové tabulce.
- **Typ skupiny (group type)**, který může být *indirect, all, select* a *fast failover*. Typ *indirect* podporuje pouze jeden balíček akcí, ale umožňuje více záznamům z tabulky toků ukazovat na společný identifikátor skupiny. Typ *all* uplatní všechny balíčky akcí ve skupině. Tento typ se používá u vícesměrového a všesměrového vysílání a paket se naklonuje pro zpracování každým balíčkem. U typu *select* se na základě početního algoritmu uplatní jeden balíček ze skupiny.
- **Počítadla (counters)**, která jsou aktualizována pokud je paket zpracován danou skupinou.
- Seznam **balíčků akcí (action buckets)**, kde každý z balíčků obsahuje soubor akcí, které mají být na paket uplatněny.

Tabulka měření

Tabulka měření umožňuje OpenFlow aplikaci implementovat omezení rychlosti a QoS (Quality of Service). V každém záznamu této tabulky se měří rychlost přiřazených paketů a je možno tuto rychlost řídit. Záznam v tabulce se skládá z následujících polí:

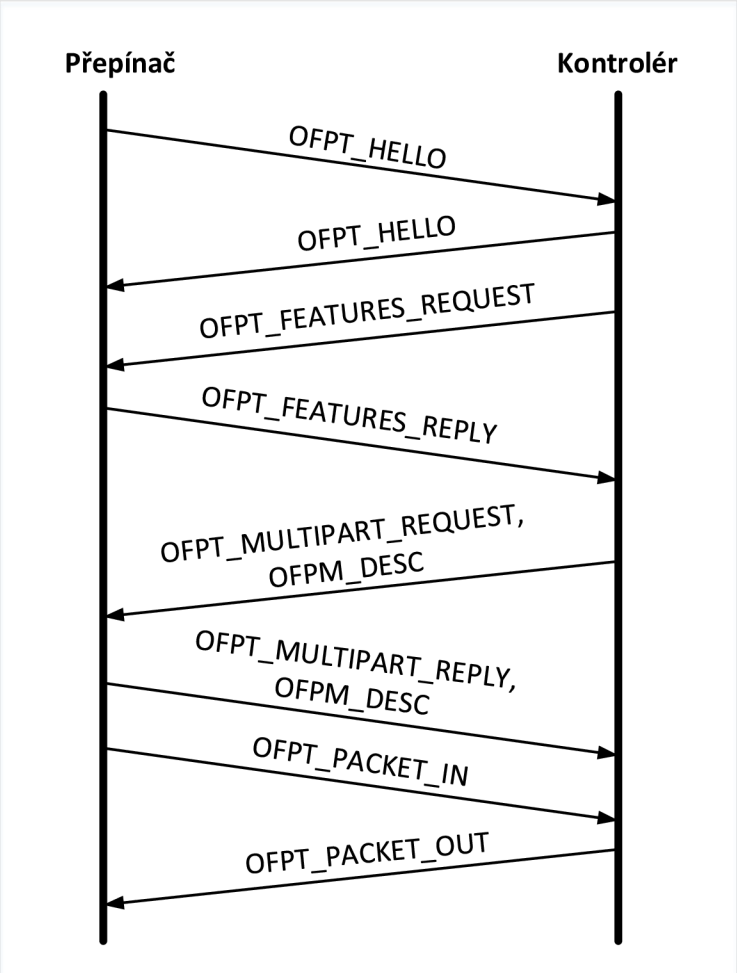
- **Identifikátor metru (meter identifier)**, což je 32 bitové celé číslo, které identifikuje měření.
- **Pásma metru (meter bands)** je seznam pásem, kde u každého pásma je specifikována rychlost a způsob zpracování přiřazeného paketu.
- **Počítadla (counters)**, která jsou aktualizována pokud je paket zpracován daným metrem.

3.2 OpenFlow protokol

Protokol OpenFlow umožňuje kontroléru přidávat, aktualizovat a mazat záznamy v tabulkách toků. Dále popisuje výměnu zpráv mezi kontrolérem a přepínači pomocí zabezpečeného kanálu.

3.2.1 Zprávy

Protokol podporuje tři typy zpráv: kontrolér přepínači, asynchronní a symetrické. Příklad komunikace mezi přepínačem a kontrolérem je na obr. 3.3



Obr. 3.3: Příklad komunikace mezi přepínačem a kontrolérem

Kontrolér přepínači

Tyto zprávy jsou iniciovány kontrolérem a slouží k přímému řízení nebo kontrole stavu přepínače a mohou vyžadovat odpověď. Níže jsou popsány některé ze zpráv tohoto typu:

- **Features** – zasláním této zprávy kontrolér požaduje od přepínače zaslaní identity a základních funkcí. Přepínač odpovídá zprávou *features reply*, kterou se identifikuje a zašle informace o svých funkcích.
- **Configuration** – kontrolér je schopen nastavit a dotazovat se na konfigurační parametry v přepínači.

- **Modify-State** – zpráva pro přidávání, odebíraní a modifikaci záznamů v tabulce toků nebo skupin a nastavování vlastností portů.
- **Read-State** – zpráva pro sběr různých informací od přepínačů, jako například aktuální konfigurace, statistiky a funkce. Většina Read-state žádostí a odpovědí jsou implementovány do tzv. vícedílních zpráv (multipart messages).
- **Packet-out** – touto zprávou může kontrolér zasílat pakety přímo ze specifického portu na přepínači a předávat pakety přijaté zprávou *Packet-in*. Zpráva *Packet-out* musí vždy obsahovat celý paket nebo ID paketu uloženého v přepínači a seznam akcí, které mají být na daný paket aplikovány ve specifikovaném pořadí. Pokud je seznam akcí prázdný, bude paket zahozen.

Asynchronní

Asynchronní zprávy jsou iniciovány přepínačem. Slouží k předání informací kontroléru o aktualizaci síťových událostí a změnách stavu přepínače. Níže jsou popsány některé z asynchronních zpráv:

- **Packet-in** – Touto zprávou převádí přepínač řízení paketu na kontrolér. Například, pokud je kontrolér v reaktivním režimu a přepínač žádá o rozhodnutí (vytvoření záznamu v tabulce toků) jak s daným paketem zacházet, nebo pokud má přepínač nastaveno pravidlo, že v případě nenalezení žádné shody, odesílá paket kontroléru. Kontrolér na tuto zprávu odpovídá zprávou *Packet-out*.
- **Flow-Removed** – informuje kontrolér o odstranění záznamu z tabulky toků. Tato zpráva je zaslána jako výsledek požadavku na smazání, nebo jako proces vypršení jednoho z časových limitů.
- **Port-status** – informace o stavu na daném portu. Zpráva může obsahovat konfiguraci portu nebo informaci o změně stavu. Například pokud uživatel manuálně vypne port nebo dojde k vyřazení linky.
- **Role-status** – informace kontroléru o jeho změně role z řídicí na podřadnou v momentě, kdy je přepínač připojen k více kontrolérům a nový kontrolér se zvolí do řídicí role.
- **Controller-Status** – přepínač tuto zprávu pošle všem kontrolérům při změně stavu přenosového kanálu, a to na kterémkoliv přepínači.

Symetrické

Symetrické zprávy jsou odesílané buď kontrolérem nebo přepínačem bez předchozího upozornění či vyžádání. Níže jsou popsány některé ze symetrických zpráv:

- **Hello** – zpráva posílána při navazování spojení.

- **Echo** – zpráva, kterou se ověřuje zda je spojení mezi kontrolérem a přepínačem aktivní. Může být také použita k měření latence a šířky pásma.
- **Error** – zpráva k upozornění na problém druhé straně připojení. Většinou používané přepínačem k indikaci selhání žádosti, která byla iniciována kontrolérem.

3.3 Verze

Protokol OpenFlow byl postupně vydán v několika verzích, které se liší podporovanými funkcemi. U OpenFlow zařízení je vždy uvedeno jakou verzi podporuje a není vždy pravidlem, že je implementována nejnovější verze. Proto je dobré znát hlavní rozdíly mezi těmito verzemi. Některé z nich jsou vypsány níže, podrobněji pak ve zdroji [11]:

- **OpenFlow 1.0** byl vydán roku 2009, byla to první oficiální podporovaná verze. Tato verze je tedy referenční, bylo v ní popsáno následující: koncept OpenFlow portu (tedy fyzického portu), tabulka toků, proces hledání shody paketu se záznamy v tabulce toků, akce a předávání paketů, tři druhy zpráv mezi kontrolérem a přepínačem apod.
- **OpenFlow 1.1** byl vydán v únoru roku 2011 a oproti předchozí verzi podporoval již více tabulek toků a skupin, specifikace virtuálního portu. Byly přidány dva módy pro vypořádání se se ztrátou konektivity, apod.
- **OpenFlow 1.2** byl vydán v květnu roku 2011 a byla přidána podpora pro IPv6, podpora více kontrolérů a jejich rolí. Byla provedena úprava zpráv Packet-in.
- **OpenFlow 1.3.0-1.3.5** byly vydány v průběhu let 2012 až 2015. Během těchto verzí nastaly tyto změny: rozšířená podpora hlaviček IPv6, přidána tabulka měření, počítadla toků, filtrování podle spojení. Byl vyhrazen port 6653 společností IANA a přidány některé další specifikace.
- V **OpenFlow 1.4.0-1.4.1** byly přidány další opisy portů a zpráv, možnosti monitorování toku v reálném čase, mnoho změn v synchronizaci a podporovaných událostech.
- V **OpenFlow 1.5.0-1.5.1** byly přidány možnosti zpracování IP paketu a PPP (Point-to-Point Protocol) rámce (dříve bylo možné pouze Ethernet), výstupní tabulky, další možnosti pro statistiky, shoda na základě TCP příznaku (SYN, ACK, FIN).

4 VIRTUALIZACE SÍŤOVÝCH FUNKCÍ

NFV je dalším moderním způsobem jak navrhovat, nasazovat a spravovat síťové služby, který se zaměřují na optimalizaci síťových služeb. Toto řešení přišlo od poskytovatelů síťových služeb, kteří potřebovali zrychlit nasazení nových síťových služeb a posílit tak svůj růst. Na trhu je velké množství proprietárního hardwaru různého druhu a mnohdy pro spuštění každé nové síťové služby je potřeba dalšího hardwarového zařízení. To je limitující pro rychlý růst, zejména kvůli náročnosti návrhu takového zařízení a rostoucích nákladů na energie. Bývá stále obtížnější umístění takového hardwaru jeho integrita a správa takto komplexního zařízení. Navíc hardwarová zařízení mají krátkou životnost, kvůli kterému je nutná častá obnova bez generování zisku. Řešení tohoto problému operátoři našli ve virtualizaci a vznikla tak skupina ISG NFV (Industry Specification Group for Network Functions Virtualization) pod ETSI (European Telecommunication Standards Institute), díky které vznikly základní požadavky a architektura NFV [6, 12].

Podstata konceptu NFV je ve využití standardní technologie virtualizace ke sjednocení mnoha typů síťových zařízení na standardní servery, přepínače a velkoobjemová úložiště, které mohou být umístěny v datacentrech, síťových uzlech a v prostorách koncového uživatele. NFV tedy odděluje software od hardwaru a poskytuje použití libovolného, komerčně dostupného hardwaru pro implementaci specifických síťových funkcí, jako je například NAT, firewall či DNS (Domain Name Service) [6].

4.1 SDN vs. NFV

Podobnost konceptů SDN a NFV tkví ve využití síťové abstrakce. SDN odděluje řídicí funkce od funkcí transportu dat, zatímco NFV se snaží tyto síťové funkce oddělit od hardwaru. Oba tyto modely silně závisí na virtualizaci, aby mohli abstrahovat infrastrukturu sítě do softwaru a poté základní software implementovat na hardwarové platformy. Již při představení konceptu NFV uvedli tvůrci tento koncept jako silně komplementární k SDN. NFV tedy může být implementován bez potřeby SDN, ale tyto dvě metody mohou být také kombinovány s vidinou vyššího potenciálu. Pomocí SDN může být zvýšen výkon, zjednodušena kompatibilita se stávajícím řešením, nebo usnadněn provoz a údržba [1].

5 DOSTUPNÁ SDN ŘEŠENÍ

V této kapitole jsou uvedeny již konkrétní řešení pro SDN. Jsou zde uvedeny open-source i komerční kontroléry, softwarové a hardwarové přepínače s podporou protokolu OpenFlow – tedy použitelné pro SDN sítě. Je zde čerpáno zejména ze zdrojů [3, 5, 4, 14, 13]

5.1 Kontroléry

Úplně prvním kontrolérem pro SDN byl NOX z roku 2008, který byl vyvíjen firmou Nicira, ale již není dále podporován.

5.1.1 Open-source kontroléry

Nejznámější a nejpoužívanější open-source kontroléry, s jejich hlavními programovacími jazyky a podporovanými protokoly jižního rozhraní, jsou vypsané v tabulce 5.1.

Tab. 5.1: Open-source kontroléry

Dostupné open-source kontroléry

Název	Jazyk	Jižní rozhraní
POX	Python	OpenFlow, OVSDB
OpenDaylight (ODL)	Java	OpenFlow
ONOS	Java	OpenFlow, BGP, OVSDB, TL1
Trema	C/C++, Ruby	OpenFlow
LOOM		OpenFlow, NETCONF
Ryu	Python	OpenFlow, OVSDB, BGP, OF-CONFIG, NETCONF
Juniper Networks Contrail	C/C++, Java, Python	BGP, XMPP, NETCONF, OVSDB
Floodlight		OpenFlow

5.1.2 Komerční kontroléry

Na trhu je dnes již celá řada komerčních kontrolérů. Jejich stručný přehled s podporou jižního rozhraní je popsán v tabulce 5.2. Vývojem SDN řešení, jako jsou kontroléry, aplikace či přepínače, se již dnes zabývají všichni významní výrobci síťových

zařízení, např. HP, Cisco, Huawei a mnoho dalších. Spousta komerčních kontrolérů vychází z některých open-source projektů.

Tab. 5.2: Komerční kontroléry

Dostupné komerční kontroléry

Název	Firma	Jižní rozhraní
SDN Fx Controller	Avaya	OpenFlow, OVSDB, OF-CONFIG, NETCONF
Big Cloud Fabric	Big Switch Networks	BGP, OpenFlow
Agility	Ciena	OpenFlow, OVSDB, BGP, NETCONF
Transcend SDN Transport Controller	Coriant	NETCONF, SNMP, TL1
OneController	Extreme Networks	OpenFlow, OVSDB, NET-CONF
Aruba VAN SDN Controller	HPE	OpenFlow, NETCONF, SNMP
Agile Controller	Huawei	OpenFlow, NETCONF, SNMP
Inocybe Open Networking Platform	Inocybe	OpenFlow, OVSDB, BGP, NETCONF
Faucet SDN Controller	Faucet	BGP, OpenFlow
B4N Controller	Brain4Net	OpenFlow
Lumina SDN Controller	Lumina Networks	BGP, Netconf, OpenFlow, OVSDB
ProgrammableFlow Controller PF6800	NEC	OpenFlow
Cisco Open SDN Controller	Cisco	BGP, Netconf, OpenFlow, OVSDB
H3C VCF Controller	H3C	OpenFlow

5.2 Přepínače

Přepínač pro SDN může mít softwarovou či hardwarovou podobu. V následujících tabulkách jsou vypsána dostupná řešení. Softwarový přepínač může být navrhnut k implementaci do hardwarového zařízení nebo pouze jako software, který slouží například ke komunikaci mezi virtuálními zařízeními. Hardwarový přepínač bývá

hybridním přepínačem, který provádí klasické přepínání a zároveň podporuje protokol OpenFlow.

5.2.1 Softwarové SDN přepínače

V tabulce 5.3 jsou vypsány některé z open-source i komerčních SDN přepínačů.

Tab. 5.3: Softwarové přepínače

Dostupné softwarové přepínače

Název	Firma	Open-source/komerční
LINC	FlowForwarding	open-source
ofsoftswitch13	Ericsson, CPqD	open-source
Open vSwitch	Open Community	open-source
Switch Light	Big Switch	komerční
Pantou/OpenWRT	Stanford	open-source
XorPlus	Pica8	komerční
Indigo	Big Switch	open-source
B4N SwitchOS	Brain4Net	komerční
Lagopus Software Switch	Lagopus	open-source

5.2.2 Hardwarové SDN přepínače

V tabulce 5.4 jsou vypsaní výrobci síťových produktů a seznam přepínačů s podporou OpenFlow.

Tab. 5.4: Hardwarové přepínače

Hardwarové přepínače s podporou OpenFlow

Firma	Produkty
Edgecore Networks	AS7712-32X, AS6812-32X, AS5812-54T, AS5712-54X, AS5710-54X, AS5600-52X
Allied Telesis	AT-x230, AT-x510, AT-x310, AT-x930 families, AT-DC2552XS
Arista	7280SE, 7050, 7050X, 7500 families

(pokračování na další straně)

(pokračování seznamu)

Firma	Produkty
Brocade	VDX 2741, VDX 6740, VDX 6940, VDX 8770, NetIron CES 2000, NetIron CER 2000, ICX 6430, ICX 6450, FCX, ICX 6610, ICX 6650, FSX 800/FSX 1600, ICX 7250, ICX 7450, ICX 7750
Centec	GoldenGate 580, V150, V330, V350
Cisco	Nexus 9300 Series, Nexus 3000 Series, Nexus 31128PQ, Nexus 3232C, Nexus 3264Q, C9500, C9300, C3850, C3650
Extreme Networks	Summit X440, Summit X460, Summit X460-G2, Summit X480, Summit X670, Summit X670-G2, Summit X770
H3C	S6800-4C, S6800-4C, S6800-2C, H3C S6800-32Q, H3CS6800-54QT, H3CS5130-54S-HI, H3CS5130-54C-PWR-HI, H3CS5130-54C-HI, H3CS5130-34C-HI, ...
HPE	2920, 3500, 3800, 5400 , 6200, 6600 , 8200, FlexNetwork 5130, FlexNetwork 7500, FlexFabric 5700 series, FlexFabric 7900, ...
Huawei	S5720, S6720-EI, S7700, S9700, S12700 series switches
Juniper Networks	EX9200, MX80, MX240, MX480, MX960, MX2010, MX2020, QFX5100
NEC	PF1000 Virtual Switch, PF5240 Switch, PF5248 Switch
Nokia	OmniSwitch 6250, 6350, 6450-10, 6855, 6860(E), 6865, 10K, 9900,...
Pica8	P3297, P3922, P3930, P5101, P5401
Quanta	BMS T3048-LY9, T3048-LY8, T3048-LY2R, T3048-LY2, T5032-LY6, T1048-LB9, T3040-LY3, T5016-LB8D

6 NÁSTROJE PRO TVORBU LABORATORNÍ ÚLOHY

Účelem laboratorních cvičení je seznámení studentů s teoretickými poznatky dané problematiky a následné prověření pomocí praktických úkolů.

Pro praktické příklady je využito virtuálního prostředí. Pro laboratorní účely je připraven virtuální stroj s operačním systémem Ubuntu (viz 6.1.2) a emulačním nástrojem Mininet (viz 6.3). Dále je využito externího kontroléru OpenDaylight (viz 6.4) a síťového analyzátoru Wireshark (viz 6.2).

6.1 Virtuální prostředí

Pro laboratorní účely je vhodné využít virtuálního prostředí. Pro laboratorní úlohu je dostačující jedno hardwarové PC, na kterém lze spustit několik virtuálních zařízení s libovolným operačním systém a dále díky emulátoru Mininet lze sestavit celou topologii sítě. Virtualizace nám tedy nahradí absenci několika hardwarových zařízení.

6.1.1 Oracle VM VirtualBox

Jako virtualizační nástroj je použit Oracle VM VirtualBox. VirtualBox je distribuován pro Windows, Linux, MacOS a Solaris. Pro hostování podporuje velké množství desktopových i serverových operačních systémů. Pro naše účely nám stačí podpora linuxové distribuce Ubuntu 16.04 LTS. VirtualBox je pod licencí General Public License version 2, díky které je software zdarma a volně šiřitelný. Instalace i tvorba virtuálního zařízení je velmi intuitivní [15].

6.1.2 Ubuntu

Ubuntu je volně šiřitelná linuxová distribuce pro pracovní stanice i servery založená na Debian GNU/Linux, jehož vývoj podporuje společnost Canonical. Je to operační systém s plně grafickým rozhraním GNOME obsahující programy pro běžnou činnost na počítači. Centrum softwaru dává možnost snadného přidávání dalších programů do počítače. Dalším způsobem jak nainstalovat program do Ubuntu je pomocí balíku – soubor s příponou `.deb` [16].

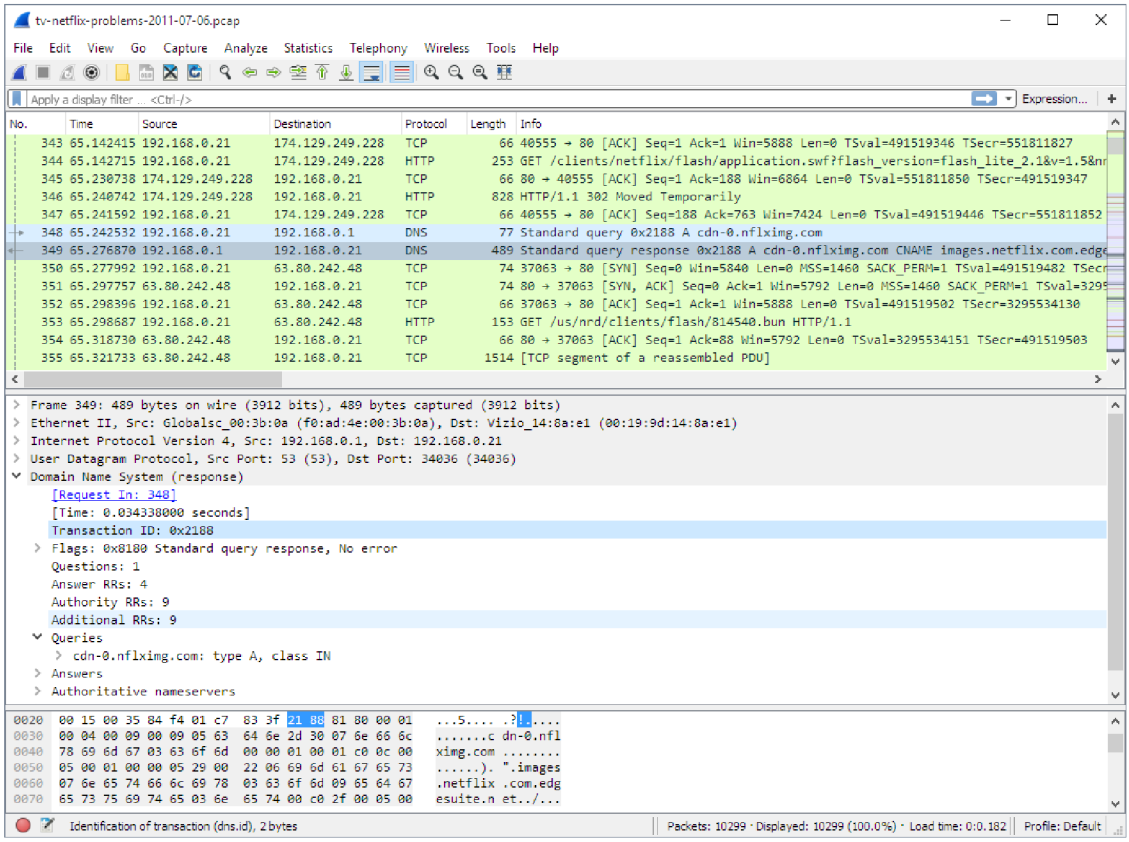
Ve virtuálním stroji pro laboratorní úlohy je nainstalován Ubuntu verze 16.10 LTS (Long Term Support), který má zajištěnou podporu až 5 let. To nám zajišťuje stabilní a bezpečný operační systém, na který můžeme instalovat síťový emulátor

Mininet, kontrolér OpenDaylignt, Wireshark a další nástroje využitelné v laboratorní úloze.

6.2 Wireshark

Wireshark je open-source protokolový analyzátor pod licencí GNU GPLv2. Umožňuje zachytávání a interaktivní prohlížení síťového provozu. Tento program se využívá zejména při odstraňování problémů v síti, ke zkoumání bezpečnostních rizik, k analýze, vývoji software a komunikačních protokolů a ke vzdělávání. Poskytuje grafické rozhraní a je dostupný pro systémy Windows, MacOs, Linux a UNIX [17].

Mezi základní funkce patří zachytávání paketových dat ze síťového rozhraní a zobrazování velmi podrobných protokolových informací, importování paketů z textových souborů, ukládání zachyceného provozu a jeho exportování do různých souborových formátů. Dále také filtrování paketů, barevné rozlišení a vytváření statistik[17]. Grafické rozhraní s ukázkou zachycených paketů a zobrazením protokolových informací lze vidět na obr. 6.1.



Obr. 6.1: Grafické rozhraní programu Wireshark se zachycenými pakety [17].

6.3 Mininet

Mininet je open-source síťový emulátor, pomocí něhož můžeme vytvářet virtuální koncové body, přepínače, kontroléry a linky na jednom Linuxovém jádře. Mininet je nástrojem pro výzkum, vývoj, testování, ladění a další úlohy, díky kterým můžeme vytvořit kompletní virtuální síť na jednom počítači. Všechny komponenty vytvořené Mininetem jsou reálné – jen místo hardwarových zařízení je použita softwarová interpretace. Ke koncovým bodům se můžeme připojit pomocí SSH (Secure Shell). Díky podpoře OpenFlow protokolu můžeme vytvářet topologie s architekturou SDN [18].

Mezi výhody patří jednoduchost – topologii můžeme vytvořit pomocí jednoho příkazu během několika sekund. Lze vytvářet topologie libovolné velikosti, spouštět reálné programy běžící na Linuxu, spouštět Mininet na pracovní stanici, serveru či virtuálním zařízení. Podporuje také skripty psané v Pythonu[2] [18].

Pro emulaci kontroléru lze využít kontroléry NOX, Ryu, OVS Controller nebo je možné se připojit ke vzdálenému kontroléru. Pro emulaci přepínačů lze využít Open vSwitch, Linux bridge a IVS (Indigo Virtual Switch). Emulované spoje představují kabelové připojení prvků, u kterých lze díky Linux Traffic Control řídit provoz na požadovanou přenosovou rychlost [18].

V laboratorní úloze je využito připojení ke vzdálenému kontroléru OpenDaylight (viz 6.4) a přepínač Open vSwitch.

6.3.1 Mininet CLI

Mininet se ovládá pomocí příkazového řádku. Spuštění emulátoru probíhá příkazem `sudo mn`. Tím se nám vytvoří `minimal` topologie s jedním kontrolérem, jedním přepínačem, dvěma hosty a linkami spojující je s přepínačem. Prvotní výpis pak vypadá následovně:

```
*** Creating network
*** Adding controller
*** Adding hosts:
h1 h2
*** Adding switches:
s1
*** Adding links:
(h1, s1) (h2, s1)
*** Configuring hosts
h1 h2
```

[2]Python je vysokoúrovňový skriptovací programovací jazyk.

```
*** Starting controller
c0
*** Starting 1 switches
s1 ...
*** Starting CLI:
```

Následně je možné vkládat příkazy pro Mininet. Příkazem `help` lze vyvolat nápovědu a souhrn použitelných příkazů. V tabulce 6.1 je sepsáno několik základních příkazů a jejich význam.

Tab. 6.1: Základní příkazy pro Mininet

Základní příkazy pro Mininet

Příkaz	Význam
nodes	zobrazí dostupné uzly
net	zobrazí spojení mezi uzly
dump	vypíše informace o všech uzlech
pingall	ping mezi všemi hosty
h1 ping h2	provede ping z hostu h1 na host h2
x ifconfig	zobrazí rozhraní a informace o nich (x nahradíme názvem zařízení – h1, s1, ...)
link s1 h1 down	vypnutí linky mezi s1 a h1
link s1 h1 up	zapnutí linky mezi přepínačem s1 a hostem h1
xterm h1	zapne příkazovou řádku hosta h1
exit	ukončení Mininetu

6.3.2 Mininet API

Mininet API je postaveno na třech úrovních: nízkoúrovňové API, API střední úrovně a vysokoúrovňové API. Níže jsou tyto úrovně popsány [18]:

- **Nízkoúrovňové API** – se skládá ze základních tříd pro uzly a linky, jako například `Host`, `Switch`, `Link` a jejich podtřídy. Používání tohoto API však není moc rozšířené. Příklad tvorby sítě s tímto API:

```
h1 = Host( 'h1' )
h2 = Host( 'h2' )
s1 = OVSSwitch( 's1', inNamespace=False )
c0 = Controller( 'c0', inNamespace=False )
Link( h1, s1 )
Link( h2, s1 )
```

```
h1.setIP( '10.1/8' )
h2.setIP( '10.2/8' )
c0.start()
s1.start( [ c0 ] )
print h1.cmd( 'ping␣-c1', h2.IP() )
s1.stop()
c0.stop()
std::cout << i << std::endl;
```

- **API střední úrovně** – Přidává objekt `Mininet`, který slouží jako kontejner pro uzly a linky. Poskytuje několik metod pro přidání uzlů a linek do sítě, jako například `addHost()`, `addSwitch()`, `addLink()`, a také metody pro síťovou konfiguraci, spuštění a vypnutí. Příklad tvorby sítě s tímto API:

```
net = Mininet()
h1 = net.addHost( 'h1' )
h2 = net.addHost( 'h2' )
s1 = net.addSwitch( 's1' )
c0 = net.addController( 'c0' )
net.addLink( h1, s1 )
net.addLink( h2, s1 )
net.start()
print h1.cmd( 'ping␣-c1', h2.IP() )
CLI( net )
net.stop()
```

- **Vysokoúrovňové API** – přidává abstrakci topologii šablon a třídu `Topo`, která poskytuje možnost vytvářet opakovatelně využitelnou a parametrizovanou šablonu topologie. Tyto šablony je možné použít pomocí příkazu `mn` s parametrem `--custom` a zadat je z příkazového řádku. Příklad tvorby sítě s tímto API:

```
class SingleSwitchTopo( Topo ):
    "Single␣Switch␣Topology"
    def build( self, count=1 ):
        hosts = [ self.addHost( 'h%d' % i )
                  for i in range( 1, count + 1 ) ]
        s1 = self.addSwitch( 's1' )
        for h in hosts:
            self.addLink( h, s1 )
```

```
net = Mininet( topo=SingleSwitchTopo( 3 ) )
net.start()
CLI( net )
net.stop()
```

6.3.3 Tvorba topologií

Mininet podporuje parametrizované topologie. S několika řádky kódu v jazyce Python lze vytvořit flexibilní topologii, která může být konfigurována a znovu použitelná pro další projekty a experimenty. Všechny třídy a struktury celého Mininet Python API lze nalézt ve zdroji [19]. Příkladem jednoduché topologie, kde je pouze jeden přepínač a k němu připojeno n koncových zařízení [18]:

```
from mininet.topo import Topo
from mininet.net import Mininet
from mininet.util import dumpNodeConnections
from mininet.log import setLogLevel

class SingleSwitchTopo(Topo):
    "Single␣switch␣connected␣to␣n␣hosts."
    def build(self, n=2):
        switch = self.addSwitch('s1')
        # Python's␣range(N)␣generates␣0..N-1
for␣h␣in␣range(n):
host␣=␣self.addHost('h\%s'␣\%␣(h␣+␣1))
self.addLink(host,␣swit)
```

Topologie lze vytvářet i v CLI pomocí jednoho příkazu jako například:

```
sudo mn --switch=ovs --controller=nox --topo=single,3
--test pingall.
```

Příkaz z uvedeného příkladu nám vytvoří topologii s jedním kontrolérem NOX, přepínačem OVSSwitch a k němu připojenými třemi hosty. Dále bude proveden test konektivity, kde se každý uzel snaží provést ping na ostatní zařízení. Mezi další volby jednořádkového příkazu pro tvorbu topologie patří například `--host, -- link, --mac, --arp` a další. Všechny části topologie a jejich možnosti lze zobrazit příkazem `mn -h` [18]. K některým částem topologie lze za zvolenou možnost vložit i parametry. Nicméně tyto parametry již v nápovědě nejsou uvedeny.

Pro výběr druhu topologie slouží příkaz `--topo`. Tím můžeme například vytvořit hvězdicovou, lineární či stromovou topologii. Nicméně pro určení prvků v síti je nutné znát základní parametry, které jsou vypsány níže:

- `--topo single,n` – vytvoří hvězdicovou topologii o jednom přepínači a n koncovými stanicemi,
- `--topo tree,depth=n,fanout=k` – vytvoří stromovou topologii o hloubce n s k koncovými stanicemi na každém přepínači,
- `--topo linear,n` – vytvoří lineární topologii s n přepínači a jednou koncovou stanicí na každém přepínači.

6.4 OpenDaylight

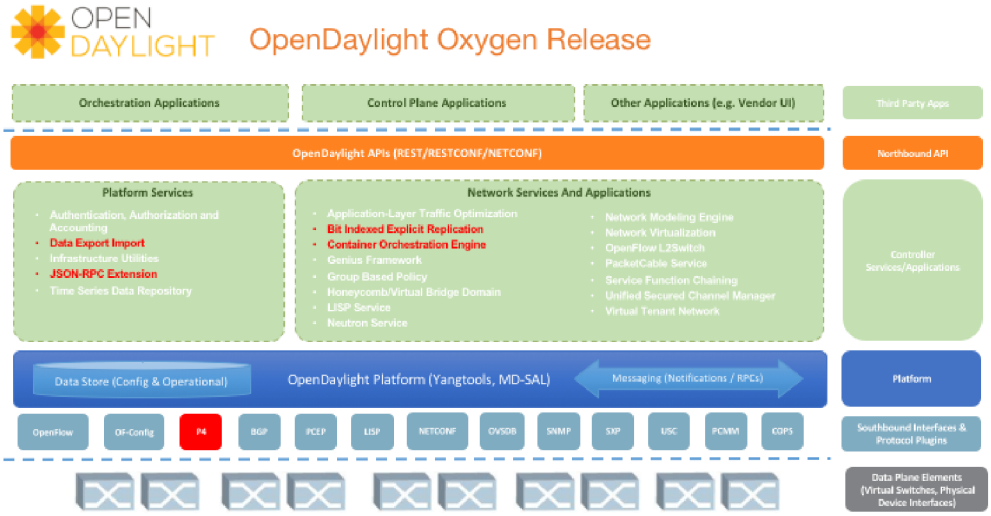
OpenDaylight je open-source projekt zaštítěný společností The Linux Foundation. Je to modulární, rozšiřitelný, škálovatelný více protokolový kontrolér pro SDN, který běží v JVM (Java Virtual Machine). Potenciálně tedy může být spuštěn z jakéhokoliv operačního systému a hardwaru podporující Javu. Lze jej využít pro centralizované monitorování, správu a orchestraci.

Architektura OpenDaylight (aktuální verze Oxygen 0. 8. 0) kontroléru je zobrazena na obr. 6.2. Kontrolér pro komunikaci s aplikacemi, jako jsou síťové aplikace, aplikace třetích stran a uživatelské rozhraní, využívá jako severní rozhraní REST API, RESTCONF a NETCONF. Samotný kontrolér poskytuje síťové služby a aplikace pro směrování, analýzu a bezpečnost a další služby pro fungování platformy. Pro jižní rozhraní mimo OpenFlow využívá i další protokoly jako například OVSDB, NetCONF, SNBI apod. , díky kterým lze připojit zařízení podporující OpenFlow, Open vSwitches a další virtuální i fyzické zařízení.

Platforma kontroléru poskytuje centralizovaně logický pohled na topologii fyzické sítě a umožňuje koordinovat změny pravidel směrování všech síťových zařízení. Využívá LLDP (Link Layer Discovery protocol) zprávy k objevení topologie OpenFlow zařízení. DLUX rozhraní kontroléru poskytuje grafický pohled (viz obr. 6.4.1) na topologii s přepínači a hosty. Správce topologie ukládá a spravuje informace o zařízeních v doméně, a to včetně jejich možností a dostupnosti. Mezi další komponenty patří správce zařízení, správce přepínačů či trekování hostů [20].

6.4.1 Ovládání

Pro nastavení síťového prostředí OpenDaylight kontroléru je, mimo instalaci základního softwaru, potřeba nainstalovat další funkce dle specifických potřeb. Součástí



Obr. 6.2: Architektura OpenDaylight kontroléru [20]

platformy OpenDaylight je Apache Karaf[3], který nám poskytuje běhové prostředí k implementaci funkcí do kontroléru. Ve výchozím nastavení nejsou nainstalované žádné funkce. Po instalaci samotného kontroléru lze instalovat jednotlivé funkce dle vlastního výběru, a to pomocí Karaf konzole bez potřeby restartu kontroléru [21].

Instalace

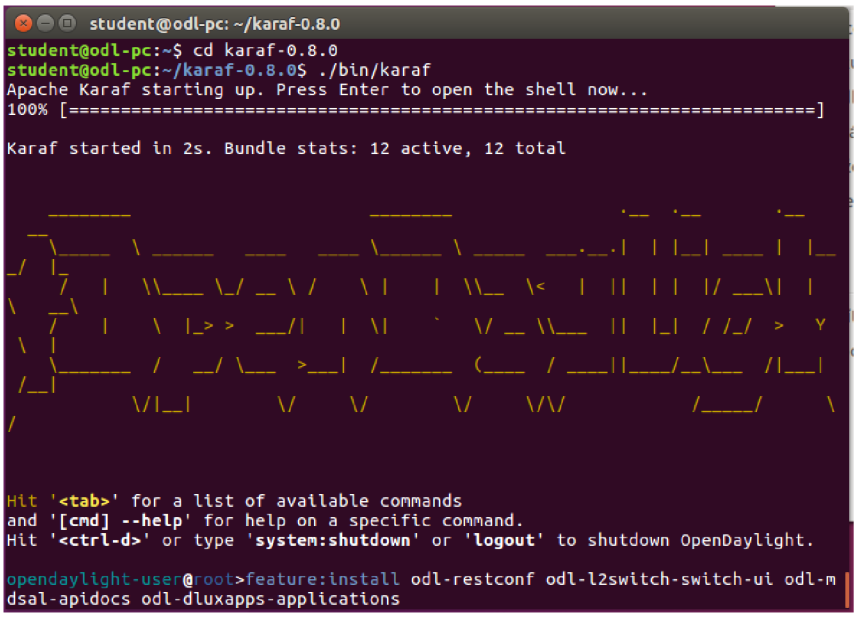
Pro instalaci kontroléru je doporučeno prostředí některé Linuxové distribuce s nainstalovaným JVM a Java JDK. Zdroj balíčků aktuálně podporovaných verzí kontroléru je dostupný na adrese `http://www.opendaylight.org/software/downloads`. OpenDaylight je zabalen do Karaf kontejneru, který umožňuje vývojářům vložit potřebný software do jediné složky [21].

Kroky pro spuštění kontroléru v rámci karaf distribuce pomocí příkazové řádky:

1. Rozbalit stažený soubor (zip/tar) – např. `tar -xvf karaf-0.8.0.tar.gz`.
2. Přejít do rozbalené složky – např. `cd karaf-0.8.0`.
3. Spustit `./bin/karaf`.

Po spuštění souboru se nám otevře Karaf konzole (obr. 6.3), ve které můžeme přidávat funkce kontroléru. Příkazem `feature:list` zobrazíme kompletní seznam dostupných funkcí, příkazem `feature:install <feature1>`, kde feature1 je název konkrétní funkce, nainstalujeme zvolenou funkci. K vypnutí kontroléru slouží kombinace tlačítek `ctrl+d` nebo příkazy `system:shutdown` a `logout` [21].

[3]Apache Karaf je runtime prostředí v jazyce Java. Poskytuje kontejner do kterého mohou být umístěny různé komponenty a aplikace.



Obr. 6.3: Konzola Karaf kontejneru a instalace funkcí

Funkce Karaf

Funkce kontroléru se instalují odděleně, dle aktuálních potřeb uživatele. V tab. 6.2 je seznam několika základních funkcí, které lze nainstalovat včetně jejich významů [21].

Práce s DLUX rozhraním

DLUX je aplikace pro správu sítě v OpenDaylight kontroléru poskytující intuitivní grafické rozhraní (viz obr 6.4).

DLUX obsahuje několik modulů, které lze pomocí Karaf konzole implementovat hromadně pomocí balíčku `odl-dluxapps-aplications` nebo odděleně v podobě balíčků `odl-dlux-core`, `odl-dluxapps-nodes`, `odl-dluxapps-topology`, `odl-dluxapps-yangui`, `odl-dluxapps-yang visualizer a odl-dluxapps-yangman`.

Ke grafickému rozhraní lze přistupovat přes webový prohlížeč zadáním adresy `http://<ipadresa-zařízení>:8181/index.html`. Výchozí přihlašovací údaje pro uživatelské jméno i heslo jsou `admin`.

Pro laboratorní účely jsou v grafickém prostředí využity balíčky pro `Topology`, `Nodes` a `Yang UI`. Položka `Topology` poskytuje grafický náhled na topologii sítě, v položce `Nodes` lze vidět výpis všech přepínačů v sítí a statistiky jejich rozhraní. Modul `Yang UI` poskytuje náhled na veškerá data a nastavení kontroléru v hierarchické struktuře.

Tab. 6.2: Základní funkce kontroléru OpenDaylight

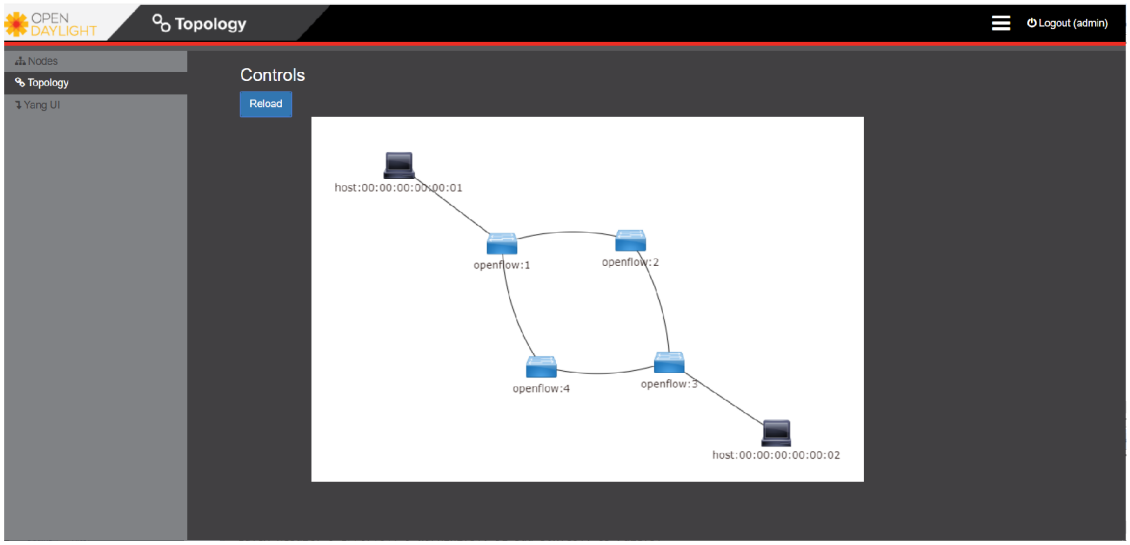
Základní funkce kontroléru OpenDaylight

Název	Popis	Příkaz v Karaf
DLUX	poskytuje intuitivní grafické rozhraní	odl-dluxapps-applications
L2 Switch	poskytuje L2 přepínání při připojení OpenFlow přepínače	odl-l2switch-switch-ui
L2 Switch - loopremover	implementace protokolu STP	odl-l2switch-loopremover
OpenFlow plugin	podporuje připojení k sítím využívající OpenFlow protokol, aktuálně podporuje verze 1. 0 a 1. 3	odl-openflowplugin-flow-services
RESTCONF API Support	přes tuto funkci mj. probíhá autentizace při přihlašování v DLUX	odl-restconf

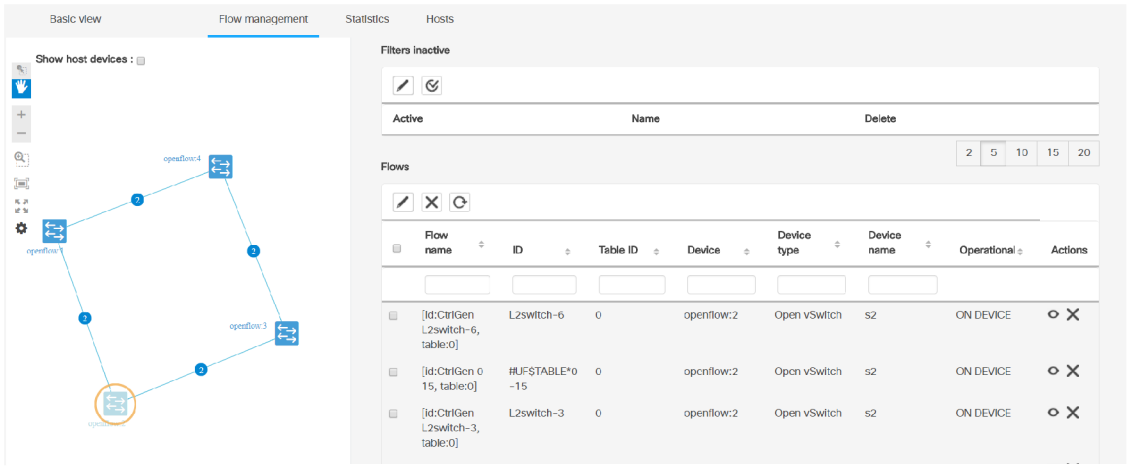
6.4.2 OpenFlow Manager

OpenFlow Manager je open-srouce SDN aplikace pro OpenDaylight kontrolér vyvinuta společností Cisco Systems. Rozšířená verze tohoto programu je součástí Cisco Open SDN Controller, což je komerční distribuce kontroléru vyvinutá z OpenDaylight kontroléru společností Cisco Systems.

Aplikace poskytuje grafické rozhraní (viz obr. 6.5) k vizualizaci topologie, intuitivnímu prohlížení, úpravám, tvorbě a odstraňování záznamů v tabulkách toků. Dále poskytuje různé statistiky k tokům a portům a souhrnné informace o koncových stanicích. K rozhraní se přistupuje přes webový prohlížeč zadáním adresy `http://<localhost>:9000`.



Obr. 6.4: Grafické rozhraní OpenDaylight kontroléru



Obr. 6.5: Grafické rozhraní aplikace OpenFlow Manager

7 POPIS LABORATORNÍ ÚLOHY

Cílem laboratorní úlohy je seznámit studenty s konceptem SDN a protokolem OpenFlow v rámci předmětu Architektura sítí. Poukázat na hlavní rozdíly mezi konceptem SDN a tradičními přístupem k návrhu datových sítí, kterými jsou především oddělené centrální řízení, způsob jakým pracuje kontrolér a přepínač, využití tabulek toků a programovatelnost.

Jelikož během studia se tento koncept ve výuce neobjevuje, bylo pro vypracování laboratorní úlohy zapotřebí se s ním detailně seznámit a najít takové softwarové vybavení, které bude vyhovovat laboratorním podmínkám a podporovat koncept i protokol OpenFlow. Software byl vybírán tak, aby laboratorní úloha nebyla pouhou simulací chování sítě, ale aby alespoň částečně reflektovala reálnou síť. I přesto, že celá síť běží v jedné virtuální stanici, jsou emulovány reálné prvky v síti ke kterým lze přistupovat. OpenDaylight je reálným kontrolérem, který lze využít v síti. Některé komerční kontroléry staví právě na tomto kontroléru.

Výzvou pro tuto práci bylo vytvořit takovou laboratorní úlohu, která objasní rozdíly mezi koncepty a pomocí jednoduchých úkolů ukáže praktickou implementaci SDN a protokolu OpenFlow. Jednoduchost tkví v použití grafických nástrojů namísto příkazových interpretů, díky kterým se student snadno orientuje v prostředí a může se soustředit na podstatu vykonávaných úkolů.

Laboratorní úloha je rozdělena na teoretickou a praktickou část. V teoretické části je popsán koncept SDN v porovnání s tradičními sítěmi, se kterými se student běžně setkává ve výuce i v běžném životě. Teorie je zaměřena na souhrn architektury tradičních sítí, o níž by měli mít studenti znalosti již z předchozího studia, dále na architekturu SDN, protokol OpenFlow (zejména architektura OpenFlow přepínače, používaných zpráv a funkce tabulek toků), simulátor Mininet a kontrolér OpenDaylight. Důraz je kladen na prvky v sítí a řízení v obou konceptech, tabulky toků a zprávy v protokolu OpenFlow a základní ovládání Mininetu.

Teoretické poznatky, které jsou uvedeny v laboratorním úvodu, jsou založeny na kapitolách 2, 3 a 6 této práce. Tento úvod by měl být nastudován ještě před zahájením plnění laboratorního cvičení. Poskytne studentovi teoretické informace, které následně prověří a uplatní v praktických úkolech.

Kromě teoretických poznatků je využito praktických příkladů, které si studenti vypracují. V praktické části se využívají virtuální nástroje využívající podporující koncept SDN a protokol OpenFlow, jako jsou například síťový simulátor Mininet, kontrolér OpenDaylight a síťový analyzátor Wireshark, které jsou popsány v kapitole 6.

K úloze je vypracován laboratorní návod (viz B) se kterým budou studenti pracovat. Návod obsahuje teoretické poznatky tématicky zaměřené na danou problema-

tiku, návod jak postupovat při plnění praktických úkolů a kontrolní otázky, které prověří získané poznatky a znalosti studenta.

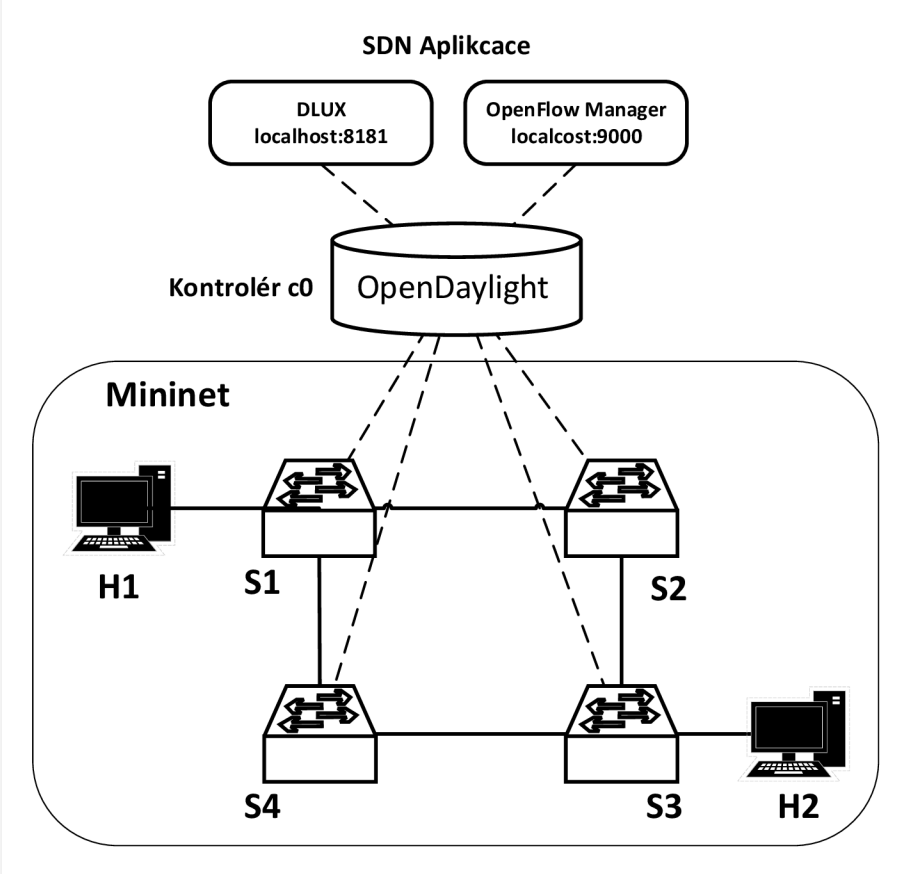
7.1 Praktická úloha

Pro vypracování praktických úkolů je připraven virtuální počítač s operačním systémem Ubuntu. Konfigurace, postup instalace a nastavení veškerého použitého softwaru je popsána v příloze A.

Úloha je členěna do několika úkolů, které jsou zaměřeny na demonstraci způsobu konfigurace a komunikace v SDN a také poukázání na odlišnosti oproti tradičnímu pojetí sítí.

Na konci každého úkolu jsou zadány kontrolní otázky, které mají prověřit získané znalosti studenta. Tyto otázky slouží i k zamyšlení nad právě provedenými úkony. Student si tedy v odpovědi shrne průnik znalostí, které získal při nastudování teoretického úvodu a provedení zadaného úkolu.

Obecné schéma zapojení laboratorní úlohy je na obr. 7.1. Toto zapojení se však mění dle aktuálně řešeného úkolu.



Obr. 7.1: Obecné schéma zapojení laboratorní úlohy

V prvním úkolu si studenti spustí virtuální počítač a vyzkouší si práci s emulátorem Mininet. Především se seznámí s výpisy, vyzkouší základní příkazy včetně nápovědy a vytvoří jednoduchou topologii.

Ve druhém úkolu si spustí síťový analyzátor Wireshark a kontrolér OpenDaylight, k jehož grafickému rozhraní se následně připojí. Kontrolér tedy není emulován Mininetem, ale je zvoleno připojení ke vzdálenému kontroléru OpenDaylight. Díky grafické nadstavbě se pomocí kontroléru zobrazí vytvořená topologie a seznam zařízení. Pomocí Wiresharku zachytí provoz mezi přepínačem a kontrolérem a analyzují zprávy protokolu OpenFlow. V tomto úkolu je vytvořena topologie pouze s jedním přepínačem, protože každý přepínač komunikuje s kontrolérem a v topologii s více přepínači, by v zachyceném provozu bylo mnoho stejných zpráv, což by vedlo k nepřehlednosti.

Cílem třetího úkolu je seznámení se s tabulkami toků a ovlivnění provozu změnami či vytvoření záznamů v kontroléru. Studenti si zobrazí tabulku toků v přepínači Open vSwitch, identifikují jednotlivá pole v záznamech a analyzují účel jednotlivých záznamů. Pro úpravu toků je použita grafická aplikace OpenFlow Manager, ke které se přistupuje přes webový prohlížeč.

Poslední úkol je zaměřen na STP protokol v SDN. Studenti si spustí topologii obsahující smyčku v síti, která je připravena ve virtuálním počítači v podobě Python skriptu. Prozkoumají, jak síť tyto smyčky řeší, zda je topologie funkční a jak se chová při výpadku linky.

8 ZÁVĚR

Diplomová práce se věnuje moderním síťovým technologiím, které využívají výhody virtualizace. Nedostatky konvenčních sítí daly motivaci pro tvorbu moderních otevřených a programovatelných sítí. Práce je zejména zaměřena na architekturu SDN, která je podrobně popsána a porovnána s architekturou tradičních sítí. Podstatou architektury sítí je oddělení řídicí a datové části, kde se řídicí část přemístila do zařízení, které nazýváme kontrolér, a datová část zůstala v zařízeních, jako jsou přepínače. Oproti tradičním sítím se tedy řízení sítě stalo centralizovaným. I koncept SDN má své limity, jako například ve škálovatelnosti a nárocích na vysoký výkon. Tyto limity lze však flexibilněji řešit a to díky programovatelnosti sítí a případným zvyšováním počtu kontrolérů.

Důležitou součástí SDN je OpenFLow, který popisuje podobu přepínače a stanovuje protokol pro komunikaci mezi kontrolérem a přepínačem. Nicméně to není jediný protokol, který zajišťuje komunikaci mezi kontrolérem a přepínačem. Mezi další protokoly patří například OVSDB či ForCES. Dalším konceptem moderních sítí je NFV, které lze s SDN kombinovat. Tento koncept není pro danou práci stěžejní a proto není v práci detailně popsán.

Koncept SDN je již velice rozšířen a všichni velcí výrobci síťových technologií pro něj vyvíjí či vyrábí různá řešení. Vzniklo i spoustu open-source projektů jako například i OpenDaylight, který je využit pro laboratorní úlohu.

V práci jsou dále popsány nástroje, které byly využity pro návrh laboratorní úlohy a se kterými budou studenti pracovat. Stěžejní je síťový emulátor Mininet, který nám dává možnost vytvořit z jednoho místa rozsáhlou virtuální síť s kontrolérem, přepínači, koncovými zařízeními a propojujícími linkami. Tento emulátor a kontrolér OpenDaylight bude spouštěn z virtuálního zařízení s operačním systémem Ubuntu 16.04 LTS, které bylo vytvořeno pomocí virtualizačního nástroje VirtualBox. Pro laboratorní úlohu byl zpracován návod a dokument pro vyučujícího. V rámci úlohy se studenti seznámí s konceptem SDN. Prakticky prověří centralizované řízení sítě, reálnou implementaci tabulek toků, změny záznamů toků a vyzkouší aplikace, které nejsou v jádru kontroléru. Dále prozkoumají možnosti emulované sítě pomocí Mininetu, analyzují zprávy definované OpenFlow protokolem a vyzkouší chování sítě, ve které je redundantní cestou vytvořena smyčka.

LITERATURA

[1] *Network Virtualization.* SDxCentral [online]. [cit. 2017-11-24]. Dostupné z URL: <https://www.sdxcentral.com/sdn/network-virtualization/>

[2] FEAMSTER, Nick, Jennifer REXFORD a Ellen ZEGURA *The road to SDN.* ACM SIGCOMM Computer Communication Review. 2014, 44(2), 87-98. DOI: 10.1145/2602204.2602219. ISSN 01464833. Dostupné z URL: <http://dl.acm.org/citation.cfm?doid=2602204.2602219>

[3] KREUTZ, Diego, Fernando M. V. RAMOS, Paulo ESTEVES VERISSIMO, Christian ESTEVE ROTHENBERG, Siamak AZODOLMOLKY a Steve UHLIG *Software-Defined Networking: A Comprehensive Survey.* Proceedings of the IEEE [online]. 2015, 103(1), 14-76 [cit. 2017-10-16]. DOI: 10.1109/JPROC.2014.2371999. ISSN 0018-9219. Dostupné z URL: <http://ieeexplore.ieee.org/document/6994333/>.

[4] GORANSSON, Paul a Chuck BLACK. *Software defined networks: a comprehensive approach.* Boston: Elsevier, Morgan Kaufmann, 2014. ISBN 978-0124166752.

[5] FARHADY, Hamid, HyunYong LEE a Akihiro NAKAO. *Software-Defined Networking: A survey. Computer Networks* [online]. 2015, 81, 79-95 [cit. 2017-10-16]. DOI: 10.1016/j.comnet.2015.02.014. ISSN 13891286. Dostupné z URL: <http://linkinghub.elsevier.com/retrieve/pii/S1389128615000614>

[6] STALLINGS, William *Foundations of Modern Networking: SDN, NFV, QoE, IoT, and Cloud.* Crawfordsville, Indiana: Addison-Wesley Professional, 2015. ISBN 9780134176024.

[7] FERNANDEZ, M. P. *Comparing OpenFlow Controller Paradigms Scalability: Reactive and Proactive.* 2013 IEEE 27th International Conference on Advanced Information Networking and Applications (AINA). IEEE, 2013, 2013, 1009-1016. DOI: 10.1109/AINA.2013.113. ISBN 978-1-4673-5550-6.

Dostupné z URL: <http://ieeexplore.ieee.org/document/6531863/>

[8] SCOTT-HAYWARD, Sandra, Gemma O'CALLAGHAN a Sakir SEZER. *Sdn Security: A Survey.* 2013 IEEE SDN for Future Networks and Services (SDN4FNS). IEEE, 2013, 1-7. DOI: 10.1109/SDN4FNS.2013.6702553. ISBN 978-1-4799-2781-4. Dostupné z URL: <http://ieeexplore.ieee.org/document/6702553/>

[9] SCOTT-HAYWARD, Sandra, Sriram NATARAJAN a Sakir SEZER. *A Survey of Security in Software Defined Networks.* IEEE Communications Surveys & Tutorials. IEEE, 2016, 18(1), 623-654. DOI: 10.1109/COMST.2015.2453114. ISSN 1553-877x. Dostupné z URL: <http://ieeexplore.ieee.org/document/7150550/>

[10] BLIAL, Othmane, Mouad BEN MAMOUN a Redouane BENAINI. *An Overview on SDN Architectures with Multiple Controllers. Journal of Computer Networks and Communications.* 2016, 1-8. DOI: 10.1155/2016/9396525. ISSN 2090-7141. Dostupné z URL: <http://www.hindawi.com/journals/jcnc/2016/9396525/>

[11] *OpenFlow Switch Specification Version 1.5.1.* Open Networking Foundation [online]. 2015 [cit. 2017-11-12]. Dostupné z URL: <https://3vf60mmveq1g8vzn48q2o71a-wpengine.netdna-ssl.com/wp-content/uploads/2014/10/openflow-switch-v1.5.1.pdf>

[12] ISG NFV. *Network Functions Virtualisation – An Introduction, Benefits, Enablers, Challenges & Call for Action.* ISG NFV white paper [online]. Darmstadt-Germany, 2012 [cit. 2017-11-29]. Dostupné z URL: <http://portal.etsi.org/NFV/NFV_White_Paper.pdf>

[13] *Special Report: OpenFlow and SDN — State of the Union.* SDxCentral [online]. SDNCentral, LLC, 955 Benecia Avenue, Sunnyvale, CA 94085 USA, 2016 [cit. 2017-11-08]. Dostupné z URL: <https://www.opennetworking.org/wp-content/uploads/2013/05/Special-Report-OpenFlow-and-SDN-State-of-the-Union-B.pdf/>

[14] *2017 Network Virtualization Report: SDN Controllers, Cloud Networking and More - Online Edition.* SDxCentral [online]. SDNCentral, LLC, 955 Benecia Avenue, Sunnyvale, CA 94085 USA, 2017 [cit. 2017-11-06]. Dostupné z URL: <https://www.sdxcentral.com/reports/2017/network-virtualization-sdn-controller/>

[15] *VirtualBox* [online]. 2017 [cit. 2017-12-11]. Dostupné z URL: <https://www.virtualbox.org/>

[16] *Ubuntu* [online]. 2017 [cit. 2017-12-11]. Dostupné z URL: <https://www.ubuntu.cz/>

[17] *Wireshark User's Guide* [online]. 2018 [cit. 2017-2-10]. Dostupné z URL: <https://www.wireshark.org/docs>

[18] *LANTZ, Bob, Nikhil HANDIGOL, Brandon HELLER a Vimal JEYAKUMAR. Introduction to Mininet.* In: GitHub [online]. 2017 [cit. 2017-11-20]. Dostupné z URL:
<https://github.com/mininet/mininet/wiki/Introduction-to-Mininet>

[19] *Mininet Python API Reference Manual* [online]. 2017 [cit. 2017-12-10]. Dostupné z URL:
<http://mininet.org/api/annotated.html>

[20] *OpenDaylight wiki* [online]. 2017 [cit. 2017-12-10]. Dostupné z URL:
<https://wiki.opendaylight.org/view/Main_Page>

[21] *OpenDaylight Documentation* [online]. 2017 [cit. 2018-04-1]. Dostupné z URL:
<http://docs.opendaylight.org/en/stable-oxygen/>

SEZNAM SYMBOLŮ, VELIČIN A ZKRATEK

API	Application Programming Interface
BGP	Border Gateway Protocol
DNS	Domain Name Service
DoS	Denial-of-Service
EIGRP	Enhanced Interior Gateway Routing Protocol
ETSI	European Telecommunication Standards Institute
ForCES	Forwarding and Control Element Separation
GUI	Graphical User Interface
IP	Internet Protocol
IVS	Indigo Virtual Switch
JVM	Java Virutal Machine
LLDP	Link Layer Discovery protocol
LTS	Long Term Support
MPLS	Multiprotocol Label Switching
ISG NFV	Industry Specification Group for Network Functions Virtualization
NFV	Network Functions Virtualization
NAT	Network Address Translator
OSPF	Open Shortest Path First
OVSDB	Open vSwitch Database Management Protocol
PPP	Point-to-Point Protocol
QoS	Quality of Service
RIP	Routing Information Protocol
REST	Representational State Transfer
SCTP	Stream Control Transmission Protocol
SDN	Software-Defined Networking
SSH	Secure Shell
TCP	Transmission Control Protocol
TLS	Transport Layer Security
UDP	User Datagram Protocol
VLAN	Virtual Local Area Network

SEZNAM PŘÍLOH

A Konfigurace softwarového vybavení laboratorní úlohy **58**
A.1 VirtualBox . . . 58
A.2 Ubuntu . . . 58
A.3 Mininet . . . 59
A.4 OpenDaylight . . . 59
A.5 OpenFlow Manager (OFM) . . . 60

B Laboratorní návod **61**
B.1 Cíl laboratorní úlohy . . . 61
B.2 Vybavení pracoviště . . . 61
B.3 Teoretický úvod . . . 61
B.3.1 Architektura tradičních sítí . . . 61
B.3.2 Architektura softwarově definovaných sítí . . . 63
B.3.3 Architektura OpenFlow . . . 66
B.3.4 OpenFlow protokol . . . 68
B.4 Postup . . . 73
B.4.1 Úkol 1 . . . 73
B.4.2 Úkol 2 . . . 74
B.4.3 Úkol 3 . . . 76
B.4.4 Úkol 4 . . . 78
B.4.5 Úklid pracoviště . . . 78
B.5 Seznam zkratek . . . 79
B.6 Doporučená literatura . . . 79

C Zdrojový kód **80**

D Obsah přiloženého CD **81**

A KONFIGURACE SOFTWAROVÉHO VYBAVENÍ LABORATORNÍ ÚLOHY

A.1 VirtualBox

VirtualBox 5.2.6 pro Windows
Zdroj instalačního média: <`https://www.virtualbox.org/wiki/Downloads`>

Konfigurace DHCP serveru

Soubor -> Host Network Manager -> Properties

- **Adresa serveru** – 192.168.1.1
- **Maska serveru** – 255.255.255.0
- **Spodní hranice** – 192.168.1.100
- **Horní hranice** – 192.168.1.150

Konfigurace virtuálního zařízení

- **Název počítače** – Mininet_PC
- **Typ** – Linux
- **Verze** – Ubuntu (64-bit)
- **Velikost paměti (RAM)** – 4096 MB
- **Pevný disk** – nový virtuální disk se souborem VDI
- **Velikost pevného disku** – pevná velikost (10GB)
- **Video paměť** – 128 MB (maximální)
- **Síť** – karta 1 – NAT, karta 2 – Síť pouze s hostem

A.2 Ubuntu

Ubuntu 16.04.3 LTS (64-bit)
Zdroj instalačního média: <`https://www.ubuntu.com/download/desktop`>

Nastavení přihlašovacích údajů

- **Název počítače** – mininet-pc
- **Uživatelské jméno** – student
- **Heslo** – student

Nastavení statické adresy rozhraní

Nastavení statické adresy hosta v termínálu:

```
\$ sudo nano /etc/netowrk/interfaces
auto enp0s8
iface enp0s8 inet static
address 192.168.1.101
netmask 255.255.255.0
network 192.168.1.0
broadcast 192.168.1.255
```

A.3 Mininet

Mininet verze 2. 2. 2
Instalace ze zdroje v terminálu:

```
\$ sudo apt install git
\$ git clone git://github.com/mininet/mininet
\$ cd mininet
\$ git tag
\$ git checkout -b 2.2.2
\$ cd ..
\$ mininet/util/install.sh -a
```

A.4 OpenDaylight

OpenDaylight verze 0.8.0 (Oxygen March 22, 2018)
Zdroj instalačního média: <https://nexus.opendaylight.org/content/repositories/public/org/opendaylight/integration/karaf/0.8.0/karaf-0.8.0.tar.gz>

Instalace Java:

```
\$ sudo apt-get update
\$ sudo apt-get install default-jre-headless
\$ export JAVA_HOME=/usr/lib/jvm/default-java
```

Nastavení prostředí JAVA_HOME:

```
\$ nano ~/.bashrc #otevřít soubor
\$ export JAVA_HOME=/usr/lib/jvm/default-java
#vložit řádek do souboru
\$ source ~/.bashrc #spustit soubor
```

Instalace OpenDaylight a modulů:

```
\$ tar -xvf karaf-0.8.0.tar.gz
\$ cd karaf-0.8.0
\$ ./bin/karaf
\$ feature:install odl-restconf-all odl-l2switch-switch-ui
 odl-mdsal-apidocs odl-dlux-core odl-dluxapps-topology
 odl-dluxapps-nodes odl-dluxapps-yangui
```

A.5 OpenFlow Manager (OFM)

Instalace OFM softwaru a dalších potřebných částí:

```
\$ sudo apt-get install -y npm
\$ sudo apt-get install -y nodejs-legacy
\$ git clone http://github.com/CiscoDevNet/OpenDaylight-
Openflow-App.git
\$ cd OpenDayLight-OpenFlow-App/
\$ sudo nano ./ofm/src/common/config/env.module.js
#změnit baseURL z localhost na ip adresu zařízení
\$ sudo npm install -g grunt-cli
\$ sudo grunt
```

B LABORATORNÍ NÁVOD

B.1 Cíl laboratorní úlohy

Cílem laboratorní úlohy je seznámení se s konceptem softwarově definovaných sítí a to zejména s architekturou SDN, rozdíly oproti konvenčním sítím, protokolem OpenFlow a softwarovými nástroji podporující tento koncept. Pomocí softwarových nástrojů nahlédnete do reálné implementace.

B.2 Vybavení pracoviště

- 1× stolní počítač s RAM min. 8GB a nainstalovaným virtualizačním softwarem VirtualBox.
- 1× obraz disku virtuální stanice Mininet_PC, který obsahuje OS Ubuntu, Mininet, OpenDaylight, OpenFlow Manager, Wireshark.
- 1× laboratorní návod.

B.3 Teoretický úvod

B.3.1 Architektura tradičních sítí

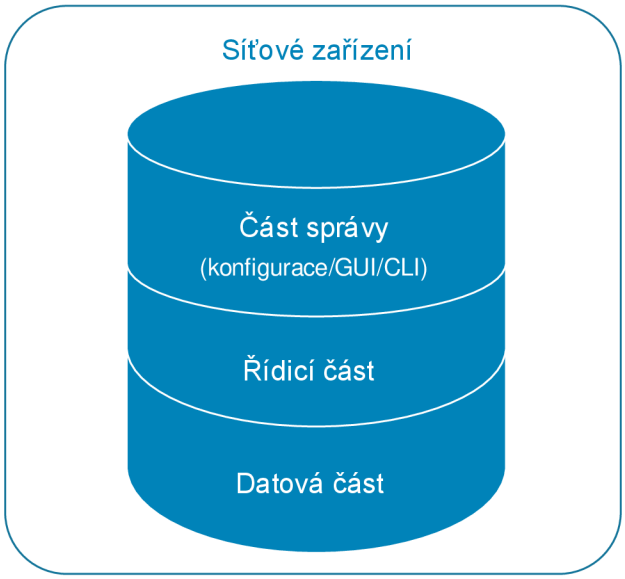
Pro lepší pochopení fungování softwarově definovaných sítí a rozdílů oproti dnes používaným sítím, je dále popsán základ architektury tradičních sítí.

Struktura síťových zařízení

Síťový prvek (např. přepínač či směrovač) je zobrazen na obr. B.1, jeho funkci lze rozdělit do tří částí: část správy, řídicí část a část transportu dat. Všechny tyto části jsou pevně implementovány v síťovém zařízení a jsou dále popsány:

- **Část transportu dat (data plane)** obsahuje porty a přepojovací pole pro příjem a odesílání paketů. Zajišťuje tedy přepojování provozu na základě rozhodnutí která stanovila řídicí část. Dále obsahuje vyrovnávací paměť paketů, plánování paketů a úpravu záhlaví. Pokud se však informace v záhlaví příchozích paketů nenachází ve směrovací tabulce nebo pokud této části nepřísluší zpracování daných paketů, je zpracování předáno řídicí části.
- **Řídicí část (control plane)** rozhoduje jak zacházet se síťovým provozem na základě definované politiky. Udržuje aktuální informace ve směrovací tabulce, aby datová část mohla odbavit co nejvíce provozu. Řídicí část je také zodpovědná za zpracování informací od různých řídicích protokolů .

- **Část správy (management plane)** obsahuje možnosti konfigurace a monitoringu. Pomocí této vrstvy architektury definujeme síťovou politiku.



Obr. B.1: Struktura síťových zařízení v tradičních sítích

Komunikace v síti

Další důležitou charakteristikou tradičních sítí je způsob předávání datových jednotek mezi zařízeními. Zde jsou popsány základní myšlenky předávání datových jednotek na 2. a 3. vrstvě (označení vrstev je dle ISO/OSI modelu):

Přepínání rámců na 2. vrstvě

Obecně přepínání na L2 může být podle identifikátoru cesty (u technologie ATM jsou to VCI/VPI) nebo na základě cílové adresy. Nejrozšířenější technologií v lokálních sítích je Ethernet. Komunikace u této technologii probíhá zasíláním rámců s využitím fyzické adresy (MAC adresy). Prvkem v síti již dnes bývá výhradně přepínač (L2 switch), který přepíná rámce na základě cílové MAC adresy uvedené v přijatém rámci. Přepínač obsahuje MAC tabulku, kde jsou uloženy MAC adresy cílových stanic a jejich cílové rozhraní (tedy informace o tom, na jaký port má být rámec s určitou cílovou adresou odeslán).

Záznamy do MAC tabulky si přepínač postupně přidává na základě procházejících rámců. U příchozího rámce si přečte cílovou adresu stanice a hledá shodu ve své tabulce. Pokud je shoda nalezena, zašle rámec na uvedený port. Pokud záznam pro tuto adresu nalezen není, zašle rámec na všechny porty mimo příchozího portu.

MAC adresu a port cílové stanice si přepínač zaznamená při odpovědi. Také si přečte zdrojovou MAC adresu stanice. Tuto adresu a port, ze kterého rámec přišel, si přidá do své tabulky. Takto se postupně učí MAC adresy a porty všech koncových stanic v síti.

Směrování paketů na 3. vrstvě

Komunikace na 3. vrstvě je označována jako směrování a využívá IP adresy. Prvek v síti, který provádí směrování je obvykle směrovač (router) nebo přepínač 3. vrstvy (L3 switch). Každý směrovač má směrovací tabulku, která obsahuje obraz topologie, podle které se rozhoduje kam daný paket zašle. Záznamy v tabulce mohou být statické (nastavené správcem sítě) nebo dynamické (vytvářené pomocí směrovacího protokolu). V obecné rovině funkce směrování obsahuje protokol pro sběr informací o topologii dané sítě, provozu, kapacitě linek a v případě dynamického směrování i algoritmus pro navrhování tras sítí.

Limity tradičních sítí

Takto hardwarově zaměřená infrastruktura má několik nedostatků, jako je obtížná správa, flexibilita, rozšiřitelnost a decentralizace. Kvůli svázanosti řídicí části a části přenosu dat je nutné konfigurovat každý síťový prvek zvlášť. Taková síť bývá náchylná na kolize a nevyváženosti zátěže. Hledání a odstraňování problémů je zde také obtížnější. Síťová zařízení obvykle podporují několik konfiguračních příkazů závislých na daném operačním systému či firmwaru, který je typicky proprietární.

B.3.2 Architektura softwarově definovaných sítí

SDN je architektura založena na fyzicky oddělené řídicí a datové části. Řídicí část již není uvnitř síťového zařízení, jako to je v tradiční architektuře, ale již na externím zařízení zvané kontrolér, které vzdáleně řídí jednu či více síťových jednotek jako jsou například přepínače a směrovače. Řízení celé sítě je tedy v této architektuře centralizované a díky SDN aplikacím, bežícím nad kontrolérem, je síť plně programovatelná.

Architektura SDN je zobrazena na obr. B.2 a skládá se z následujících částí:

- V **datové části** pracují přepojovací zařízení, která přenášejí a zpracovávají data na základě rozhodnutí od řídicí části – tedy kontroléru. Zařízení tedy komunikuje s kontrolérem a je jím řízeno. Základní charakteristikou síťových zařízení pracující v této části jsou funkce jednoduchého přeposílání bez možnosti vlastního autonomního rozhodování.

- **Jižní rozhraní** poskytuje logické propojení mezi SDN kontrolérem a síťovým prvkem operující v datové části (jako např. přepínač, virtuální přepínač, směrovač, firewall, atd.). Nejrozšířenějším API (Application Programming Interface) pro jižní rozhraní je protokol OpenFlow.
- **Řídicí část** je v SDN sítích implementována jako jeden či více kontrolérů běžících na serverech či virtuálních serverech. Kontrolér sbírá směrovací informace od SDN přepínačů pro sestavení nejlepší cesty. Dále také přijímá a zpracovává aplikační události, zajišťuje bezpečnostní mechanismy, vytváří a udržuje informace o topologii přepínačů, sbírá informace o provozu procházející danou sítí a konfiguruje parametry a atributy přepínače.
- **Severní rozhraní** umožňuje aplikacím třetích stran přístup k funkcím a službám řídicí části.
- **Aplikační část** obsahuje aplikace a služby, které definují, upravují a řídí chování a prostředky dané sítě. Díky těmto aplikacím, které mohou být aplikacemi třetích stran, je síť plně programovatelná a lze jimi rozšířit stávající funkce sítě. Mohou být aplikace jako například firewall, monitorovací systém či GUI (Graphical User Interface) pro správu a konfiguraci kontroléru, přepínačů a aplikací.

Komponenty

Mezi základní komponenty, které jsou graficky popsány na obr. B.3, patří SDN síťové zařízení, kontrolér a aplikace.

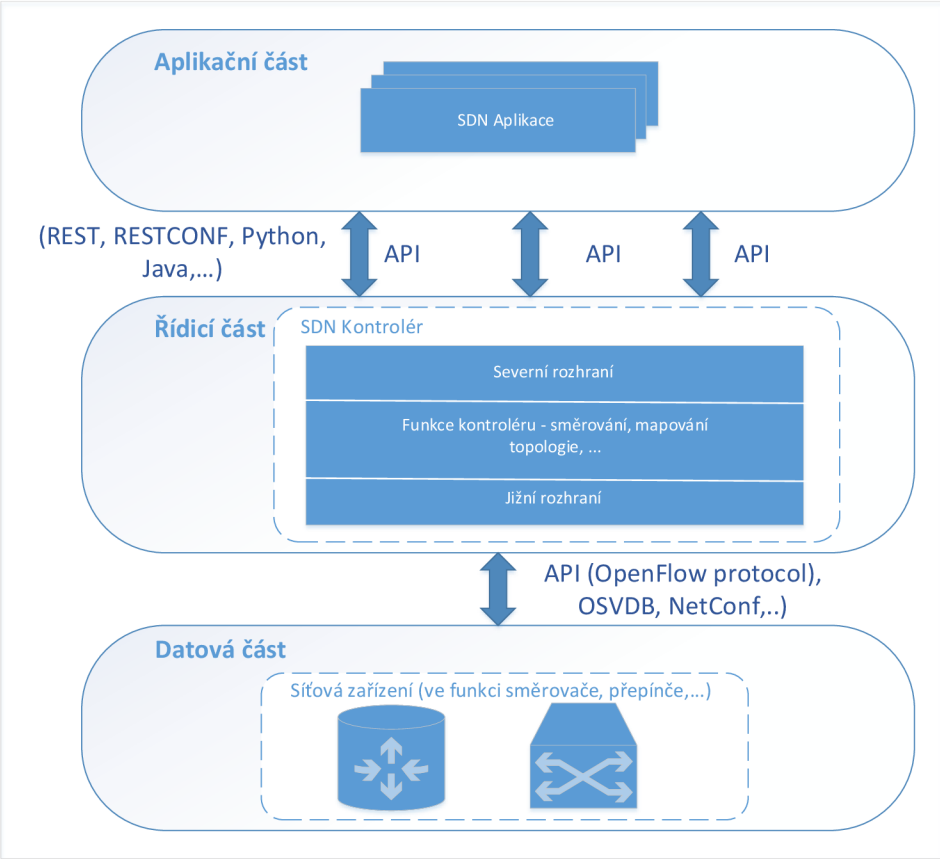
SDN síťové zařízení

Tato zařízení pracují v datové rovině. Kontrolér definuje tok (flow), který je popsán jako sada paketů přenášených z jednoho koncového bodu na druhý. Stejná pravidla se použijí na všechny pakety patřící do jednoho toku. Pravidla se v podobě záznamů (flow entry) vloží do tabulky toků (flow table), která je v přepínači. Na základě takto aplikovaných pravidel se SDN přepínač může chovat jako směrovač, přepínač, firewall, NAT (Network Address Translator), nebo jako kombinace zmíněných.

Kontrolér

Kontrolér udržuje přehled o celé síti, řídí všechna SDN zařízení v dané síti a poskytuje jižní rozhraní pro aplikace. Vkládáním záznamů do tabulek toků přepínačů implementujte pravidla týkající se směrování, přeposílání a vyvažování zátěže.

Jak je patrno z obr. B.2, kontrolér se skládá z několika modulů – rozhraní pro severní a jižní API a jádro s několika aplikacemi bežící mezi těmito rozhraními. Mezi základní funkce jádra kontroléru patří:



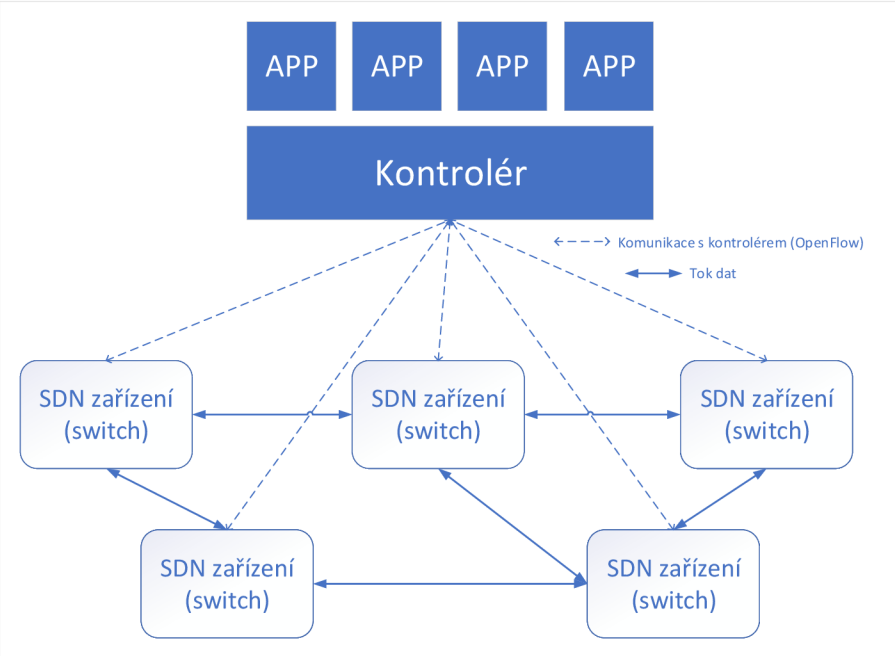
Obr. B.2: Architektura softwarově definovaných sítí

Kontrolér vytváří záznamy pro veškerý provoz v síti. Existují dva režimy pro vkládání záznamů toků do přepínače – reaktivní a proaktivní.

- V **reaktivní režimu** kontrolér zpočátku neimplementuje žádná pravidla, která by určovala jak zacházet s příchozími rámci. Při přijetí rámce se přepínač dotáže kontroléru jak daný rámec zpracovat. Kontrolér na základě informací o topologii rozhodne, jak daný rámec zpracovat a vytvoří záznam v tabulce toků. Tento záznam platí pro všechny rámce stejného typu. Přepínač se tedy dotazuje na každý rámec, u kterého nenašel shodu v tabulce toků – tím se tabulka postupně plní záznamy.
- U **proaktivního režimu** kontrolér vytvoří záznamy v tabulkách toků na základě informací o topologii pro všechny možné toky ještě před zahájením provozu v síti. V tomto režimu se generuje méně zpráv pro kontrolér.

Komunikace v SDN sítích

V SDN sítích jsou funkce přepojování centralizované uvnitř kontroléru, který má ucelené informace o stavu celé sítě. Díky těmto informacím může sestavit nejkratší



Obr. B.3: Komponenty SDN a komunikace mezi nimi

cestu a může aplikovat směrovací politiku stanovenou od SDN aplikace. SDN přepínače fungující v datové části jsou oproštěny od zpracovatelské a úložné zátěže.

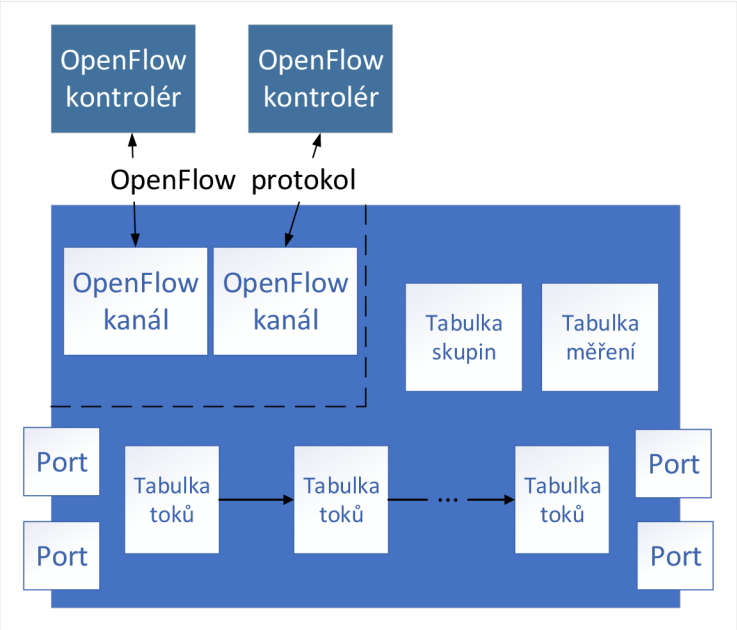
B.3.3 Architektura OpenFlow

OpenFlow definuje způsob komunikace mezi kontrolérem a přepínači. Dále také určuje jak by zařízení měla reagovat v různých situacích a jak by měla reagovat na příkazy od kontroléru. Obsahuje sadu zpráv, které jsou odesílány v obou směrech.

OpenFlow přepínač

OpenFlow přepínač je zobrazen v obr. B.4 a obsahuje jednu či více tabulek toků, tabulku skupin a tabulku měření. Dále obsahuje jeden či více OpenFlow kanálů propojující přepínač a kontrolér, přes který spolu komunikují pomocí OpenFlow protokolu. OpenFlow také definuje tři druhy portů: fyzické, logické a rezervované.

Funkcí tohoto přepínače je po přijetí paketu hledání shody v tabulce toků a na základě priority nalezené shody provést příslušnou akci. Paket může být poslán dál z lokálního portu s případnou úpravou některých polí v hlavičce, zahozen či předán kontroléru přes zabezpečený kanál. Pokud není nalezna žádná shoda, pak záleží na konfiguraci – paket může být například zahozen, poslán kontroléru či pokračovat na další tabulku toků. Veškerá pravidla a toky jsou nastavována kontrolérem.



Obr. B.4: Skladba OpenFlow přepínače [11]

Tabulky toků

Každý paket, který přijde do přepínače, je porovnáván se záznamy v jedné či více tabulek toků. Každá tabulka obsahuje několik záznamů. Struktura záznamů je zobrazena na obr. B.5.

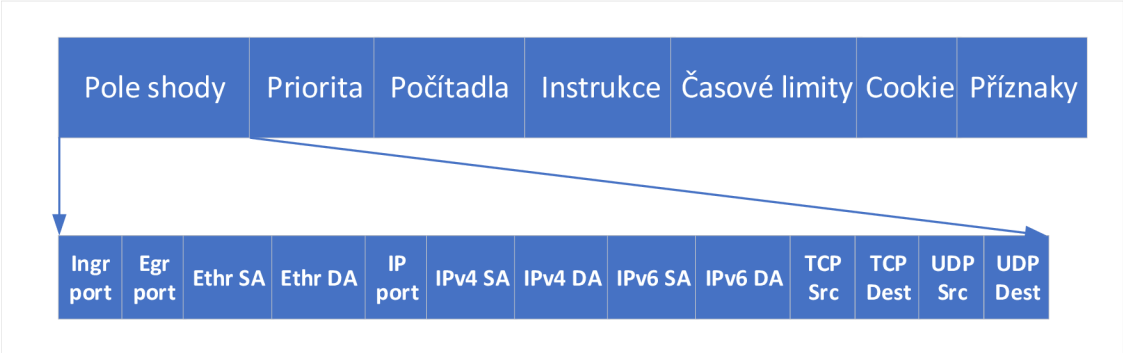
Pokud je tabulek více, jsou zřetězeny. Tabulky jsou očíslovány, z nichž první tabulka začíná hodnotou 0. Při hledání shody se operuje nejprve s Tabulkou 0 a na základě výsledku mohou být využity i další tabulky. Při nalezení shody paketu se záznamem v tabulce toků může být instrukcí tohoto záznamu odeslání paketu do jiné tabulky s vyšší hodnotou. Například může být záznam v Tabulce 0, který na základně určité IPv4 adresy odešle pakety dál ke zpracování do Tabulky 1, kde již bude shoda na základně protokolu transportní vrstvy a tudíž bude jiný záznam pro protokol TCP a jiný pro UDP (User Datagram Protocol).

Jednotlivá pole tabulky jsou popsána následovně:

- **Pole shody (match fields)** se využívá k výběru paketů, které se shodují v hodnotách tohoto pole. Shoda může být provedena, jak je vidět na obr. 3.2, na základě několika povinných hodnot. A to například na vstupním či výstupním portu přepínače, ethernetové adrese, verzi IP, IP adrese či TCP/UDP portech. Nicméně mohou být využívany i další pole, pokud jsou podporovány OpenFlow přepínačem. Mohou to být například VLAN ID, Tunnel ID či SCTP (Stream Control Transmission Protocol) port.
- Relativní **priorita (priority)** záznamu. Je to 16ti bitové pole, kde 0 repre-

zentuje nejnižší hodnotu.

- **Počítadla (counters)** se aktualizují, když je nalezena shoda s paketem.
- **Instrukce (instructions)**, které se mají provést pokud nastane shoda. Například předání na výstupní port či zahození.
- **Časové limity (timeoutes)** jsou definovány jako maximální doba před vypršením toku a způsobují odstranění záznamu z tabulky po určitém čase (hard_timeout) nebo po čase, kdy není nalezena shoda se žádným paketem (idle_timeout).
- **Cookie** je 64ti bitová hodnota určená kontrolérem. Může být použita kontrolérem k filtrování statistik toku, ke změně či smazání toku. Tato hodnota se nevyužívá při zpracování paketu.
- **Příznaky (flags)** mění způsob řízení záznamů tabulky.



Obr. B.5: Struktura záznamů v OpenFlow tabulce [6]

B.3.4 OpenFlow protokol

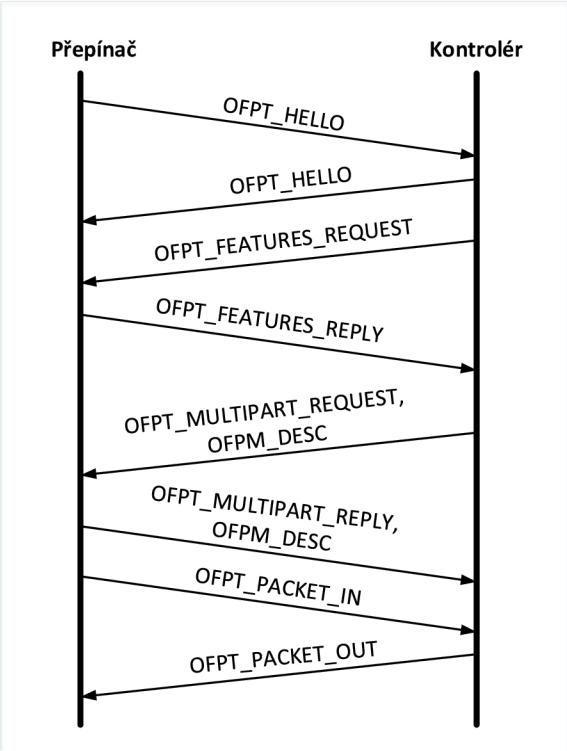
Protokol OpenFlow umožňuje kontroléru přidávat, aktualizovat a mazat záznamy v tabulkách toků. Dále popisuje výměnu zpráv mezi kontrolérem a přepínači pomocí zabezpečeného kanálu.

Zprávy

Protokol podporuje tři typy zpráv: kontrolér přepínači, asynchronní a symetrické. Příklad komunikace mezi přepínačem a kontrolérem je na obr. B.6.

Kontrolér přepínači

Tyto zprávy jsou iniciovány kontrolérem a slouží k přímému řízení nebo kontrole stavu přepínače a mohou vyžadovat odpověď. Níže jsou popsány některé ze zpráv tohoto typu:



Obr. B.6: Příklad komunikace mezi přepínačem a kontrolérem

- **Features** – zasláním této zprávy kontrolér požaduje od přepínače zaslaní identity a základních funkcí. Přepínač na tuto zprávu odpovídá zprávou *features reply*, kterou se identifikuje a zašle informace o svých funkcích.
- **Configuration** – kontrolér je schopen nastavit a dotazovat se na konfigurační parametry v přepínači.
- **Modify-State** – zpráva pro přidávání, odebíraní a modifikaci záznamů v tabulce toků nebo skupin a nastavování vlastností portů.
- **Read-State** – zpráva pro sběr různých informací od přepínačů, jako například aktuální konfigurace, statistiky a funkce. Většina Read-state žádostí a odpovědí jsou implementovány do tzv. vícedílních zpráv (multipart messages).
- **Packet-out** – touto zprávou může kontrolér zasílat pakety přímo ze specifického portu na přepínači a předávat pakety přijaté zprávou *Packet-in*. Zpráva *Packet-out* musí vždy obsahovat celý paket nebo ID paketu uloženého v přepínači a seznam akcí, které mají být na daný paket aplikovány ve specifikovaném pořadí. Pokud je seznam akcí prázdný, bude paket zahozen.

Asynchronní

Asynchronní zprávy jsou iniciovány přepínačem a slouží k předání informací kontroléru o aktualizaci síťových událostí a změnách stavu přepínače. Níže jsou popsány

některé z asynchronních zpráv:

- **Packet-in** – touto zprávou převádí přepínač řízení paketu na kontrolér. Například pokud je kontrolér v reaktivním režimu a přepínač žádá o rozhodnutí (vytvoření záznamu v tabulce toků), jak s daným paketem zacházet, nebo pokud má přepínač nastaveno pravidlo, že v případě nenalezení žádné shody, odesílá paket kontroléru. Kontrolér na tuto zprávu odpovídá zprávou *Packet-out*.
- **Flow-Removed** – informuje kontrolér o odstranění záznamu z tabulky toků. Tato zpráva je zaslána jako výsledek požadavku na smazání nebo jako proces vypršení jednoho z časových limitů.
- **Port-status** – informace o stavu na daném portu. Zpráva může obsahovat konfiguraci portu nebo informaci o změně stavu. Například pokud uživatel manuálně vypne port, nebo k vyřazení linky.
- **Role-status** – informace kontroléru o jeho změně role z řídicí na podřadnou v momentě, kdy je přepínač připojen k více kontrolérům a nový kontrolér se zvolí do řídicí role.
- **Controller-Status** – přepínač tuto zprávu pošle všem kontrolérům při změně stavu přenosového kanálu na kterémkoliv přepínači.

Symetrické

Symetrické zprávy jsou odesílané buď kontrolérem nebo přepínačem bez předchozího upozornění či vyžádání. Níže jsou popsány některé ze symetrických zpráv:

- **Hello** – zpráva posílána při navazování spojení.
- **Echo** – zpráva, kterou se ověřuje zda je spojení mezi kontrolérem a přepínačem aktivní. Může být také použita k měření latence a šířky pásma.
- **Error** – zpráva k upozornění na problém druhé straně připojení. Většinou používané přepínačem k indikaci selhání žádosti, která byla iniciována kontrolérem.

Mininet

Mininet je open-source síťový emulátor, pomocí něhož můžeme vytvářet virtuální koncové body, přepínače, kontroléry a linky na jednom Linuxovém jádře. Mininet je nástrojem pro výzkum, vývoj, testování, ladění a další úlohy, díky kterým můžeme vytvořit kompletní virtuální síť na jednom počítači. Všechny komponenty vytvořené Mininetem jsou reálné – jen místo hardwarových zařízení je použita softwarová interpretace. Ke koncovým bodům se můžeme připojit pomocí SSH (Secure Shell). Díky podpoře OpenFlow protokolu můžeme vytvářet topologie s architekturou SDN.

Pro emulaci kontroléru lze využít kontroléry NOX, Ryu, OVS Controller nebo je možné se připojit ke vzdálenému kontroléru. Pro emulaci přepínačů lze využít Open vSwitch, Linux bridge a IVS (Indigo Virtual Switch). Emulované linky představují kabelové připojení prvků, u kterých lze díky Linux Traffic Control řídit provoz na požadovanou přenosovou rychlost.

V laboratorní úloze je využito připojení ke vzdálenému kontroléru OpenDaylight a přepínač Open vSwitch.

Mininet CLI

Mininet se ovládá pomocí příkazového řádku. Spuštění emulátoru probíhá příkazem `sudo mn`. Tím se nám vytvoří `minimal` topologie s jedním kontrolérem, jedním přepínačem, dvěma hosty a linkami spojující je s přepínačem. Následně je možné vkládat příkazy pro Mininet. Příkazem `help` lze vyvolat nápovědu a souhrn použitelných příkazů. V tabulce B.1 je sepsáno několik základních příkazů a jejich význam.

Tab. B.1: Základní příkazy pro Mininet

Základní příkazy pro Mininet

Příkaz	Význam
nodes	zobrazí dostupné uzly
net	zobrazí spojení mezi uzly
dump	vypíše informace o všech uzlech
pingall	ping mezi všemi hosty
h1 ping h2	provede ping z hostu h1 na host h2
x ifconfig	zobrazí rozhraní a informace o nich (x nahradíme názvem zařízení – h1, s1, ...)
link s1 h1 down	vypnutí linky mezi s1 a h1
link s1 h1 up	zapnutí linky mezi přepínačem s1 a hostem h1
xterm h1	zapne příkazovou řádku hosta h1
exit	ukončení Mininetu

Tvorba topologií

Mininet podporuje parametrizované topologie. S několika řádky kódu v jazyce Python lze vytvořit flexibilní topologii, která může být konfigurována a znovu použitelná pro další projekty a experimenty. Topologie lze vytvářet i v CLI pomocí jednoho příkazu jako například:

```
sudo mn --switch=ovs --controller=nox --topo=single,3
--test pingall.
```

Příkaz z uvedeného příkladu nám vytvoří topologii s jedním kontrolérem NOX, přepínačem OVSSwitch a k němu připojenými třemi hosty. Dále bude proveden test konektivity, kde se každý uzel snaží provést ping na ostatní zařízení. Mezi další volby jednořádkového příkazu pro tvorbu topologie patří například `--host, -- link, --mac, --arp` a další. Všechny části topologie a jejich možnosti lze zobrazit příkazem `mn -h` [18]. K některým částem topologie lze za zvolenou možnost vložit i parametry. Nicméně tyto parametry již v nápovědě nejsou uvedeny.

Pro výběr druhu topologie slouží příkaz `--topo`. Tím můžeme například vytvořit hvězdicovou, lineární či stromovou topologii. Nicméně pro určení prvků v síti je nutné znát základní parametry, které jsou vypsány níže:

- `--topo single,n` – vytvoří hvězdicovou topologii o jednom přepínači a n koncovými stanicemi,
- `--topo tree,depth=n,fanout=k` – vytvoří stromovou topologii o hloubce n s k koncovými stanicemi na každém přepínači,
- `--topo linear,n` – vytvoří lineární topologii s n přepínači a jednou koncovou stanicí na každém přepínači.

OpenDaylight

OpenDaylight je open-source projekt zaštítěný společností The Linux Foundation. Je to modulární, rozšiřitelný, škálovatelný více protokolový kontrolér pro SDN, který běží v JVM (Java Virtual Machine). Potenciálně tedy může být spuštěn z jakéhokoliv operačního systému a hardwaru podporující Javu. Lze jej využít pro centralizované monitorování, správu a orchestraci.

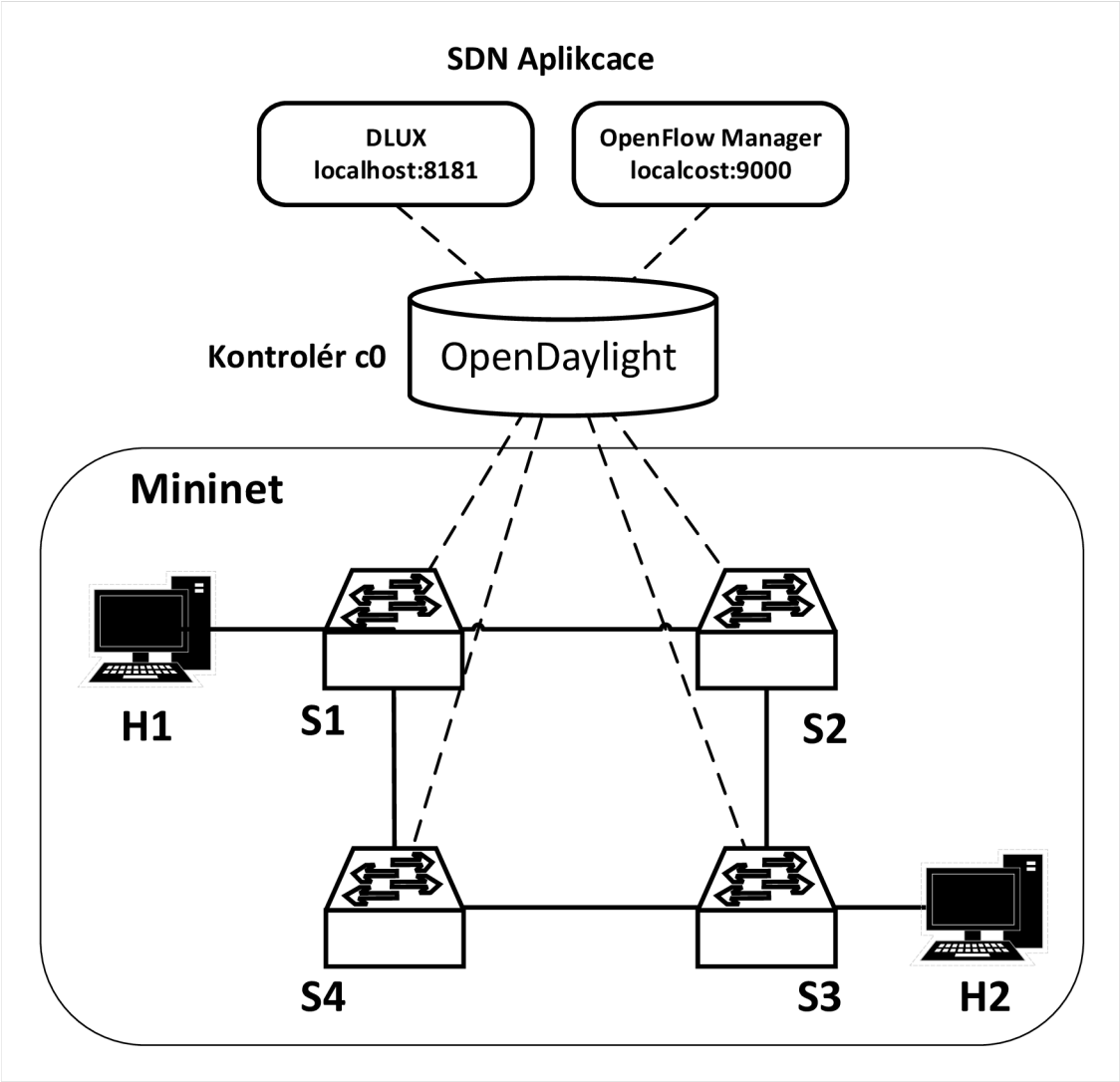
Platforma kontroléru poskytuje centralizovaně logický pohled na topologii fyzické sítě a umožňuje koordinovat změny pravidel směrování všech síťových zařízení. Využívá LLDP (Link Layer Discovery protocol) zprávy k objevení topologie OpenFlow zařízení. DLUX rozhraní kontroléru poskytuje grafický pohled na topologii s přepínači a hosty. Správce topologie ukládá a spravuje informace o zařízeních v doméně, včetně jejich funkcí a dostupnosti. Mezi další komponenty patří správce zařízení, správce přepínačů či trekování hostů.

OpenFlow Manager

OpenFlow Manager je open-srouce SDN aplikace pro OpenDaylight kontrolér vyvinuta společností Cisco Systems. Aplikace poskytuje grafické rozhraní k vizualizaci topologie, intuitivnímu prohlížení, úpravám, tvorbě a odstraňování záznamů v tabulkách toků. Dále poskytuje různé statistiky k tokům a portům a souhrnné informace o koncových stanicích. K rozhraní se přistupuje přes webový prohlížeč zadáním adresy `http://<localhost>:9000`.

B.4 Postup

Obecné schéma zapojení laboratorní úlohy je na obr. B.7. Toto zapojení se však mění dle aktuálně řešeného úkolu.



Obr. B.7: Obecné schéma zapojení laboratorní úlohy

B.4.1 Úkol 1

V prvním úkolu si spusťte virtuální počítač a seznamte se s prostředím emulátoru Mininet, vytvořte jednoduchou topologii a zjistěte informace o síti.

Postup řešení

1. Ve VirtualBox spusťte připravený virtuální stroj s OS Ubuntu, který najdete pod názvem `Mininet_PC`.
2. V Terminálu zadejte příkaz `sudo mn -h`. Prozkoumejte možnosti tvorby parametrizované topologie v emulátoru Mininet.
3. Vytvořte jednoduchou topologii podle parametrů, které jste našli v předchozím výpisu. Například příkazem `sudo mn --topo=linear,2 --controller=default --switch=ovs` vytvoříte topologii dvou přepínačů Open vSwitch propojenými mezi sebou, ke kterým je připojen kontrolér a ke každému přepínači jedna koncová stanice (obr. B.8). Vyzkoušejte i jiné topologie. Pozn.: Open vSwitch je softwarový přepínač, u kterého lze vytvořit počet portů dle aktuální potřeby. Lze využít i hardwarový přepínač s různým počtem fyzických portů (8, 24, 48, ...)
4. V mininetu vyzkoušejte příkazy: `help, nodes, dump, net a xterm h1`.
5. Emulovanou síť ukončíte příkazem `exit`.

Kontrolní otázky:

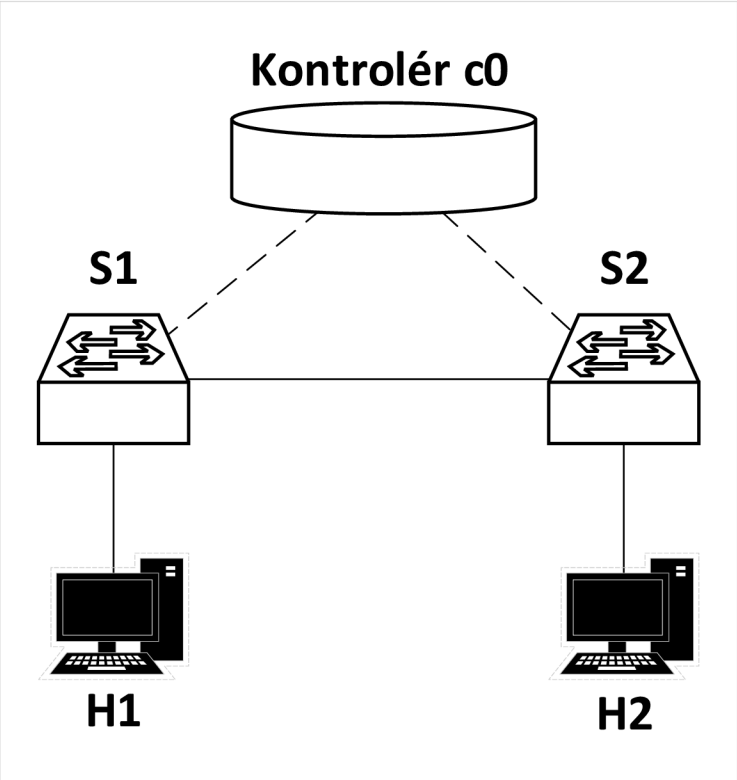
1. Jaké kontroléry a přepínače lze do topologie vložit?
2. Jaké informace získáte příkazy `node, dump, a net`?

B.4.2 Úkol 2

Ve druhém úkolu si spusťte síťový analyzátor Wireshark a kontrolér OpenDaylight. Přes webový prohlížeč se připojte ke grafickému rozhraní kontroléru. V mininetu vytvořte topologii s jedním přepínačem, který se připojí ke kontroléru OpenDaylight. Pomocí analyzátoru Wireshark zachyťte provoz a analyzujte zprávy protokolu OpenFlow.

Postup řešení

1. V novém terminálovém okně spusťte kontrolér. Nejprve přejděte do kořenové složky kontroléru příkazem `cd karaf-0.8.0` a kontrolér spusťte příkazem `./bin/karaf`. Pozn.: kontrolér může běžet i na jiném zařízení v síti, než na kterém je mininet. Je pouze nutné zařídit konektivitu mezi těmito zařízeními. Port pro kontrolér je 6653.
2. Ke grafickému rozhraní kontroléru se připojte z webového prohlížeče hostujícího či virtuálního počítače zadáním adresy



Obr. B.8: Úkol 1: Topologie vytvořena příkazem v mininet.

`http://192.168.1.101:8181/index.html`. Přihlašovací údaje jsou admin/admin. Pozn. : grafické rozhraní není součástí jádra kontroléru. Je to SDN aplikace, která ke kontroléru přistupuje přes severní rozhraní. Port této aplikace je 8181.

3. V novém terminálovém okně spusťte wireshark příkazem `sudo wireshark` a spusťte zachytávání pro rozhraní loopback.
4. V mininetu vytvořte následující topologii: `sudo mn --topo=single,3 --controller=remote,ip=192.168.1.101 --switch=ovs`. V tomto případě se vytvoří topologie s jedním přepínačem a třemi hosty. Není použit kontrolér z nabídky Mininetu, ale připojili jsme se k OpenDaylight kontroléru, který jsme si spustili v předchozím kroku.
5. V menu GUI kontroléru vyberte položku `Topology` a stiskněte tlačítko `Reload`. Nyní vidíte topologii přepínačů. Stanice sami od sebe negenerují žádný OpenFlow provoz a kontrolér je nevidí. Zadejte tedy příkaz v mininet `pingall`. Opět obnovte zobrazení topologie. Po najetí kurzoru na zařízení nebo linku uvidíte základní informace. Zápis u linek je ve tvaru `openflow:ID_zařízení:port`. Pozn. : je nutné si uvědomit, že důvod proč vidíte graficky celou topologii je ten, že v SDN je kontrolér centrálním prvkem, který sbírá informace o celé topologii a díky tomu může celou síť řídit.

6. V položce `Node` vidíte výpis všech přepínačů. Dále v Node Connectors vidíte veškeré porty daného přepínače a statistiky přenesených dat.
7. Ve Wiresharku si vyfiltrujte provoz protokolu `openflow_v4` a analyzujte zprávy.

Kontrolní otázky:

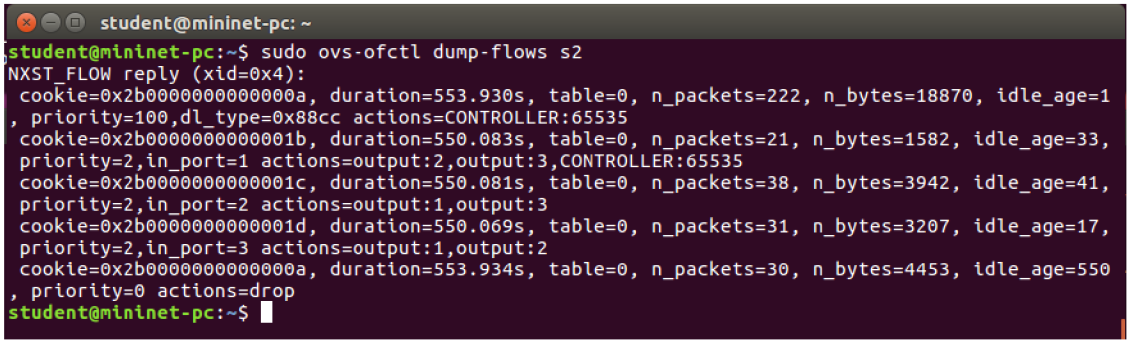
1. Jaké zprávy se ve výpisu zobrazí a k čemu slouží?
2. Ve které zprávě se přenáší popis přepínače a popis portů?

B.4.3 Úkol 3

V tomto úkolu si zobrazte tabulky toků. Identifikujte jednotlivá pole a význam všech záznamů. Pomocí SDN aplikace OpenFlow Manager vytvořte nové záznamy a ovlivněte provoz.

Postup řešení

1. V Mininetu vytvořte následující topologii: `sudo mn --topo=linear,3 --controller=remote,ip=192.168.1.101 --switch=ovs --mac`. Topologii opět uvidíte v GUI kontroléru.
2. Kontrolér je v proaktivním režimu a na základě informací o topologii vytváří pravidla pro jednotlivé přepínače, která udávají jakým způsobem má daný přepínač zacházet s příchozím provozem. Přepínač má tedy jednu či více tabulek toků se záznamy. V topologii jsou využity přepínače Open vSwitch – v terminálovém okně mininetu zobrazíte tabulku toků přepínače S2 příkazem: `sh ovs-ofctl dump-flows s2`. Tabulka toků by měla vypadat přibližně jako na obr. B.9. Pro zobrazení stanic v topologii proveďte příkaz `pingall`.



Obr. B.9: Úkol 3: Tabulka toků přepínače S2.

V tabulce je 5 záznamů s následující strukturou:

- `cookie` – identifikátor toku
- `duration` – vyjadřuje dobu od vytvoření záznamu

- `table` – identifikátor tabulky
- `n_packets` – počet datových jednotek, které byly zpracovány daným záznamem
- `n_bytes` – velikost dat, která byla zpracována daným záznamem
- `idle_age` – vyjadřuje, jak dlouho daným tokem neprošla žádná datová jednotka
- `priority` – priorita daného záznamu, jeden rámec může splňovat shodu s více záznamy, zpracován je však záznamem s nejvyšší prioritou
- `match` – pole shody, v naší tabulce se jedná o typ rámce `dl_type` a vstupní port `in_port`
- `action` – akce, která se aplikuje na datovou jednotku u které se našla shoda

První záznam udává, že pokud bude příchozí rámec typu 0x88cc (což je hodnota pro protokol LLDP, který zjišťuje topologii v síti), bude odeslán přímo kontroléru. Další tři záznamy spočívají v přeposlání příchozího provozu na další porty a mají nižší prioritu než první záznam. Poslední záznam má nulovou prioritu a nemá uvedenou žádnou shodu (match). To znamená, že veškerý provoz, pro který není uplatněn ani jeden z předchozích záznamů, je zahozen.

3. V dalším kroku zapněte aplikaci OpenFlow Manager, díky které lze v kontroléru snadno upravovat a vkládat záznamy do tabulek toků. V dalším terminálovém okně přejděte do složky aplikace příkazem `cd OpenDaylight-OpenFlow-App`. Dále zadejte příkaz `sudo grunt`. K aplikaci přistupte z webového prohlížeče hostujícího počítače zadáním adresy `http://192.168.1.101:9000`.
4. V tomto programu také vidíte grafický náhled na topologii. Přejděte na záložku `Flow Management`, kde jsou záznamy všech přepínačů. Prohlídněte si detaily některých záznamů.
5. Vytvořte nový záznam tak, aby stanice h2 nemohla komunikovat se stanicí h1. Prověřte příkazem v mininet `h2 ping h1`.
6. Po dokončení tvorby záznamů, předveďte vyučujícímu a ukončete mininet příkazem `exit`.

Kontrolní otázky:

1. Jaké shody a akce lze využít při vytváření záznamů do tabulky toků?
2. Jak se liší záznamy v tabulce toků oproti MAC tabulce v klasickém přepínači? Na základě jakých informací se hledá shoda v obou konceptech?

B.4.4 Úkol 4

Poslední úkol je zaměřen na STP protokol v SDN. Spusťte si topologii obsahující smyčku v síti, která je připravena ve virtuálním počítači. Prozkoumejte jak síť tyto smyčky odstraňuje, zda je topologie funkční a jak se chová při výpadku linky.

1. Zadejte příkaz `cd mininet/custom`. Spusťte předem připravenou topologii příkazem `sudo mn --custom=topo-lab.py --topo mytopo --controller =remote --mac`. Topologii si opět zobrazte v GUI kontoléru.
2. V topologii je smyčka, která je nežádoucí. Kontrolér má v sobě implementován STP protokol, který tyto smyčky odstraňuje. Způsob odstranění smyčky v síti probíhá vyřazením portů, které svým spojením vytvářejí redundantní trasy tak, aby byla všechna zařízení k dispozici. V závislosti na velikosti sítě může STP konvergovat delší dobu (kontrolér si udělá přehled o topologii a následně odstraní smyčky), proto je nutné vyčkat než budou všechny prvky v síti dostupné. Kontrolér označuje aktivní porty jako `forwarding` a neaktivní porty `discarding`. V GUI kontroléru otevřete položku `Yang UI` a rozbalte nabídku. Vyhledejte `opendaylight-inventory->operationals->nodes` a klikněte na tlačítko `Send`. Vypíšou se Vám informace o všech zařízeních a jejich portech. Postupně projděte všechny zařízení a jejich porty. Zjistěte, které jsou funkční a které vyřazené. Na základě těchto informací zjistěte, která linka je odpojena.
3. Vyzkoušejte příkaz `pingall`. Pokud je STP funkční, měl by ping fungovat. Při nefunkčnosti tohoto protokolu by se síť brzy zahltila a ping by nefungoval.
4. Některou z funkčních linek v Mininetu vypněte (např. `link s1 s2 down`). Ověřte zda vyřazené porty přešli do funkčního stavu. Ověřte konektivitu zařízení.
5. Opět linku zapněte a ověřte funkčnost sítě.

Kontrolní otázky:

- Proč jsou smyčky v síti nežádoucí?

B.4.5 Úklid pracoviště

- Ukončete Mininet příkazem `exit`
- Ukončete kontrolér OpenDaylight příkazem `logout`
- Ukončete webový server grunt kombinací Ctrl+C.
- Vypněte virtuální stanici.

B.5 Seznam zkratek

API	Application Programming Interface
IVS	Indigo Virtual Switch
JVM	Java Virtual Machine
LLDP	Link Layer Discovery protocol
NAT	Network Address Translator
SCTP	Stream Control Transmission Protocol
SDN	Software Defined Network
SSH	Secure Shell
STP	Spanning Tree protocol
TCP	Transmission Control Protocol
UDP	User Datagram Protocol)

B.6 Doporučená literatura

STALLINGS, William *Foundations of Modern Networking: SDN, NFV, QoE, IoT, and Cloud.* Crawfordsville, Indiana: Addison-Wesley Professional, 2015. ISBN 9780134176024.

OpenFlow Switch Specification Version 1.5.1. Open Networking Foundation [online]. 2015 Dostupné z URL: <`https://3vf60mmveq1g8vzn48q2o71a-wpengine.netdna-ssl.com/wp-content/uploads/2014/10/openflow-switch-v1.5.1.pdf`>

MORAVCOVÁ, Klára. *Tvorba virtuálních síťových topologií pomocí softwarově definovaných sítí.* Brno, 2018, 79 s. Diplomová práce. Vysoké učení technické v Brně, Fakulta elektrotechniky a komunikačních technologií, Ústav telekomunikací. Vedoucí práce: doc. Ing. Vít Novotný, Ph.D.

C ZDROJOVÝ KÓD

Zdrojový kód topologie k úloze 4 – `topo-lab.py`:

```
"""Topologie␣pro␣laboratorni␣ulohu

Adding␣the␣'topos'␣dict␣with␣a␣key/value␣pair␣to␣generate
our␣newly␣defined␣topology␣enables␣one␣to␣pass␣in
'--topo=mytopo'␣from␣the␣command␣line.
"""

from mininet.topo import Topo

class MyTopo( Topo ):
    "Simple␣topology␣example."

    def __init__( self ):
        "Create␣custom␣topo."

        # Initialize topology
        Topo.__init__( self )

        # Add hosts and switches
        h1 = self.addHost( 'h1' )
        h2 = self.addHost( 'h2' )
    s1 = self.addSwitch( 's1' )
    s2 = self.addSwitch( 's2' )
        s3 = self.addSwitch( 's3' )
        s4 = self.addSwitch( 's4' )

        # Add links
        self.addLink( s1, s2 )
        self.addLink( s2, s3 )
        self.addLink( s3, s4 )
    self.addLink( s4, s1 )
    self.addLink( h1, s1 )
    self.addLink( h2, s3 )

topos = { 'mytopo': ( lambda: MyTopo() ) }
```

D OBSAH PŘILOŽENÉHO CD

Přiložený disk CD-ROM obsahuje následující soubory a adresáře:

- **elektronická verze diplomové práce** – ve formátu pdf
- **obrázky** – adresář zdrojových obrázků ve formátu vsdx
- **laboratorní návod** – ve formátu pdf
- **navod pro přípravu pracoviště** – ve formátu pdf
- **topo-lab2.py** – Python skript s připravenou topologií k úloze 4