Univerzita Pardubice
Fakulta elektrotechniky a informatiky

Testování OpenFlow funkcionalit na hardwarových prvcích

Bc. Ondřej Šanko

Diplomová práce

2019

Univerzita Pardubice
Fakulta elektrotechniky a informatiky
Akademický rok: **2018/2019**

ZADÁNÍ DIPLOMOVÉ PRÁCE

(PROJEKTU, UMĚLECKÉHO DÍLA, UMĚLECKÉHO VÝKONU)

Jméno a příjmení:	**Bc. Ondřej Šanko**
Osobní číslo:	**I17222**
Studijní program:	**N2646 Informační technologie**
Studijní obor:	**Informační technologie**
Název tématu:	**Testování OpenFlow funkcionalit na hardwarových prvcích**
Zadávající katedra:	**Katedra softwarových technologií**

Zásady pro vypracování:

Cílem diplomové práce je popsat základní funkce protokolu OpenFlow a postup konfigurace na přepínačích Aruba. Tento postup bude obsahovat vysvětlení všech konfiguračních voleb včetně zabezpečené komunikace, debugovacích možností a nastavení komunikace s vybranými SDN kontroléry. Praktická část práce bude zaměřena na testování výkonnostních charakteristik přepínačů Aruba v OpenFlow módu. Součástí testování bude porovnání hardwarových a softwarových flow tabulek z hlediska podporovaných funkcionalit, jejich kapacit a výkonu. Ten bude testován s využitím nástrojů ICMP, iPerf a NetFlow.

Rozsah grafických prací:

Rozsah pracovní zprávy: **50**

Forma zpracování diplomové práce: **tištěná/elektronická**

Seznam odborné literatury:

OPEN NETWORKING FOUNDATION. OpenFlow Switch Specification: Version 1.5.1 (Protocol version 0x06) [online]. 2015 [cit. 2018-09-21]. Dostupné z: https://www.opennetworking.org/wp-content/uploads/2014/10/openflow-switch-v1.5.1.pdf

HP Switch Software OpenFlow Administrator's Guide K/KA/WB 15.14 [online]. 2013 [cit. 2018-09-21]. Dostupné z: http://h20628.www2.hp.com/km-ext/kmcsdirect/emr_na-c03991489-1.pdf

NADEAU, Thomas D a Kenneth GRAY. SDN: software defined networks. Sebastopol, CA: O'Reilly Media, 2013. ISBN 978-1-4493-4230-2

GORANSSON, Paul a Chuck BLACK. Software defined networks: a comprehensive approach. Amsterdam: Morgan Kaufmann, c2014. ISBN 978-0-12-416675-2

Vedoucí diplomové práce: **Ing. Filip Holík, Ph.D.**
Katedra informačních technologií

Datum zadání diplomové práce: **22. října 2018**

Termín odevzdání diplomové práce: **18. května 2019**

Univerzita Pardubice Česká republika

L.S.

Ing. Zdeněk Němec, Ph.D.
děkan

prof. Ing. Antonín Kavička, Ph.D.
vedoucí katedry

V Pardubicích dne 17. listopadu 2018

Prohlašuji:

Tuto práci jsem vypracoval samostatně. Veškeré literární prameny a informace, které jsem v práci využil, jsou uvedeny v seznamu použité literatury.

Byl jsem seznámen s tím, že se na moji práci vztahují práva a povinnosti vyplývající ze zákona č. 121/2000 Sb., autorský zákon, zejména se skutečností, že Univerzita Pardubice má právo na uzavření licenční smlouvy o užití této práce jako školního díla podle § 60 odst. 1 autorského zákona, a s tím, že pokud dojde k užití této práce mnou nebo bude poskytnuta licence o užití jinému subjektu, je Univerzita Pardubice oprávněna ode mne požadovat přiměřený příspěvek na úhradu nákladů, které na vytvoření díla vynaložila, a to podle okolností až do jejich skutečné výše.

Beru na vědomí, že v souladu s § 47b zákona č. 111/1998 Sb., o vysokých školách a o změně a doplnění dalších zákonů (zákon o vysokých školách), ve znění pozdějších předpisů, a směrnicí Univerzity Pardubice č. 9/2012, bude práce zveřejněna v Univerzitní knihovně a prostřednictvím Digitální knihovny Univerzity Pardubice.

V Pardubicích dne 22. 8. 2019

Ondřej Šanko

PODĚKOVÁNÍ

Děkuji vedoucímu práce Ing. Filipu Holíkovi, Ph.D. za cenné rady a odborné vedení, které mi poskytnul. Také bych chtěl poděkovat své rodině za podporu a trpělivost při studiích.

ANOTACE

Tato práce popisuje testování protokolu OpenFlow na fyzických prvcích společnosti HPE. V teoretické části je popsané chování protokolu OpenFlow, včetně jeho základních komponent a struktur.

V praktické části je navržena síť, na které jsou testovány funkcionality protokolu OpenFlow. Zejména se jedná o režimy, kdy přepínač ukládá pravidla do hardwarové, nebo softwarové Flow tabulky.

KLÍČOVÁ SLOVA

OpenFlow, Ryu, kontrolér, přepínač, směrovač, HPE Aruba, MikroTik

TITLE

Testing OpenFlow functionality on hardware components.

ANNOTATION

This thesis describes testing of OpenFlow protocol on hardware device from HPE. The theoretical part describes the behavior of OpenFlow protocol, including its basic components and structures.

In the practical part is designed a network on which the functionality of OpenFlow protocol is tested. In particular, the modes where the switch saves the rules in the hardware or software Flow tables.

KEYWORDS

OpenFlow, Ryu, controller, switch, router, HPE Aruba, MikroTik

OBSAH

SEZNAM OBRÁZKŮ

SEZNAM TABULEK

SEZNAM ZKRATEK

ASIC	Application Specific Integrated Circuit
HPE	Hewlett Packard
HW	Hardware
LLDP	Link Layer Discovery Protocol
ONF	Open Networking Foundation
SDN	Software-Defined Networking
SW	Software

ÚVOD

Tato práce se zabývá konceptem zvaným Softwarově definované sítě, který se stává v moderních sítích stále větším trendem. Jedná se o model, kdy skupina zařízení, je řízená jedním centralizovaným kontrolérem. Tento kontrolér definuje takzvaná Flow pravidla, které ukládá do jednotlivých přepínačů. Přepínače mají pro Flow pravidla implementovanou strukturu zvanou Flow tabulka. Jeden přepínač může mít několik těchto tabulek (neplatí pro první verzi protokolu OpenFlow). V základu se rozlišují 2 typy. Existují softwarové a hardwarové tabulky.

O implementaci tohoto abstraktního konceptu se stará protokol OpenFlow, který je popsán v teoretické části. Jsou zde popsány základní pojmy jako OpenFlow přepínač, Flow pravidla, nebo výše zmiňované Flow tabulky. Kromě pojmů je čtenář uveden také do základního fungování protokolu OpenFlow. Je obeznámen se zpracováním příchozí zprávy pomocí Pipeline processingu, nebo jaký je rozdíl mezi Ingress a Egress processingem. Dozví se, jak kontrolér navazuje komunikaci s SDN přepínačem, jaké typy zpráv jsou k tomu využity a jak zaručit zabezpečenou komunikaci pro snížení možnosti zneužití. Dále jsou popsány možnosti zabezpečení pro případy výpadku sítě, jako je výpadek kontroléru, linky, nebo celého SDN přepínače.

Praktická část je zaměřena na zapojení, konfiguraci protokolu OpenFlow na přepínači Aruba od společnosti HPE a konfiguraci kontroléru Ryu ve virtualizovaném prostředí Linux.

Dále je provedeno testování výkonnostních charakteristik. Zejména se jedná o testování výkonu protokolu OpenFlow v režimu, kdy přepínač ukládá pravidla pouze do softwarové, nebo pouze do hardwarové tabulky. Toto testování je provedeno ze dvou důvodů. První z nich je že oficiální dokumentace společnosti HPE pro zařízení Aruba neuvádí detailní specifikaci využití hardwarových, či softwarových tabulek. A to jak jejich velikosti (kolik Flow pravidel lze vložit), tak ani výkonnostní charakteristiky (propustnost komunikace, latence, či doba vkládání jednotlivých pravidel). Dalším důvodem je, že ani neuvádí, pro jaký druh konfigurace se hodí určitý druh komunikace (například zda softwarové tabulky jsou dostačující pro běžný druh komunikace, aby při nasazení do reálného provozu, třeba v datových centrech, nedocházelo k neočekávaným omezením).

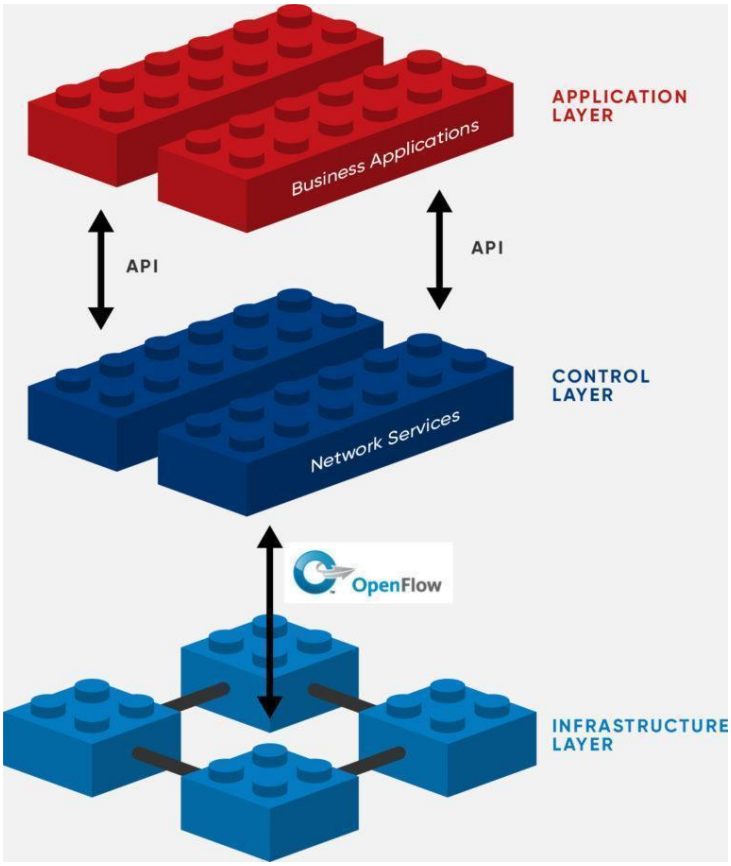
K testování jsou využity měřící nástroje, jako protokol ICMP, nástroj iPerf3, či měřící nástroje, které nabízí přepínač od společnosti MikroTik. Mezi tyto nástroje patří hlavně protokol TrafficFlow a možnost sledování přenosu na jednotlivých portech.

Čtenář bude seznámen, jak tyto nástroje použít a jak získat podrobné výsledky. V případě problému je zde uvedena kapitola, jak na přepínači od společnosti HPE provést troubleshooting a debugging.

Na závěr je provedeno vyhodnocení testů, které určí výhody a nevýhody různé konfigurace protokolu OpenFlow, včetně doporučení, pro jaký druh komunikace lze využít určitý druh konfigurace.

1 SOFTWARE-DEFINED NETWORKING

Software-defined networking (dále SDN) je architektura, která zpravidla obsahuje 3 vrstvy: aplikační vrstvu, kontrolní vrstvu a infrastrukturní vrstvu. Hlavní myšlenka, proč využíváme SDN, je právě možnost pracovat s centralizovanou kontrolní vrstvou, která „spravuje“ infrastrukturní vrstvu (také známá jako forwarding plane). (Software-Defined Networking (SDN) Definition, 2019)



Obrázek 1: SDN architektura

Zdroj: (*Open Networking Foundation, 2015*)

1.1 Aplikační vrstva

Aplikační vrstva obsahuje typicky sítové aplikace nebo funkce využívané pro správu a řízení sítě, jako je například firewall, load balancing a podobné. SDN aplikace mohou poskytovat širokou škálu nástrojů pro firmy či data centra. Společnost HPE například nabízí aplikace: HPE Network Optimizer, HPE Network protector, nebo HPE Network visualizer. (Software-Defined Networking (SDN) Definition, 2019)

1.2 Kontrolní vrstva

Kontrolní vrstva je nejdůležitější vrstvou v SDN architektuře. Reprezentuje ji takzvaný kontrolér. Kontrolér běží zpravidla na serveru a spravuje chování sítě. Definuje takzvaná Flow pravidla, která určují chování celé sítě. Moderní kontrolér je zpravidla plně programovatelný modul, v moderním vysokoúrovňovém jazyce (například JAVA v projektu FloodLight). V SDN architektuře bývá zpravidla jeden kontrolér na celou řadu přepínačů. Díky tomuto konceptu se velice usnadňuje správa sítí. Nemusí se konfigurovat každý přepínač zvlášť, ale pouze kontrolér, který spravuje všechny jemu přiřazené přepínače. (Software-Defined Networking (SDN) Definition, 2019)

1.3 Infrastrukturní vrstva

Infrastrukturní vrstvu představují zařízení, jenž zajišťují komunikaci. Mohou to být jak fyzické, tak virtualizované zařízení, které řídí kontrolér. Nejběžnější konfigurace bývá že, pravidla, podle kterých se přepínače rozhodují, co s daným tokem, přebírají od kontrolérů. V případě, že nemají k dispozici pro příchozí zprávu pravidlo, zažádá o něj kontrolér. V případě, že ani v tomto případě žádné pravidlo neodpovídá, pak záleží na konfiguraci. Může se například zahodit, zaslat na vyhrazený port a podobně.

Toto však není jediná možná konfigurace. Přepínač se například nemusí vůbec na pravidlo dotazovat, ale může rovnou zprávu zahodit, nebo kontrolér může definovat určitá pravidla, aniž by měl přepínač možnost žádat o nové. (Software-Defined Networking (SDN) Definition, 2019)

1.4 Výhody SDN

Mezi největší výhody SDN architektury rozhodně patří centralizace správy sítě. Centralizovaný kontrolér poskytuje veliké výhody jak v již zmíněné správy sítě, tak i v možnostech konfigurace. Jak již bylo uvedeno v kapitole popisující kontrolní vrstvu (1.2 Kontrolní vrstva), tak kontrolér je zpravidla plně programovatelný modul, který bývá implementován ve vysokoúrovňovém jazyce jako je například JAVA.

SDN podporuje kromě správy fyzických zařízení i správu virtuálních zařízení z centrálního kontroléru.

Centralizace poskytuje i další výhodu, jako je lepší zabezpečení. Centralizovaná správa poskytuje daleko jednodušší možnosti zabezpečení, oproti běžné sítě, kde je třeba většinou každé zařízení konfigurovat zvlášť. (Software-Defined Networking (SDN) Definition, 2019)

2 OPENFLOW

OpenFlow je považován za jeden z prvních standardů SDN. Jedná se tedy o síť, která má oddělenou kontrolní vrstvu od infrastrukturní vrstvy a zajišťuje mezi těmito vrstvami komunikaci. OpenFlow protokol je spravován neziskovou organizací Open Networking Foundation (ONF).

Mezi základní komponenty patří OpenFlow kontrolér a OpenFlow přepínač. OpenFlow přepínač zpracovává zprávy pomocí pravidel, které jsou uložené ve struktuře s názvem Flow tabulka. Pravidla přepínač přebírá od kontroléru. Základní funkcionalita je tedy taková, že na OpenFlow přepínač přijde zpráva. Pokud přepínač nemá odpovídající pravidlo, které by určilo, co se zprávou podniknout, dotáže se kontroléru o nové Flow pravidlo, pomocí předem definovaného defaultního pravidla. Přepínač si následně pravidlo uloží do své Flow tabulky a podle daného pravidla provede se zprávou akci[1]. V následující kapitolách je popsané, jak pracuje OpenFlow přepínač, jak pracuje s jednotlivými pravidly, jak pravidla vypadají, jak je „obsloužen" příchozí zprávu. Kromě principů, jak Openflow přepínač pracuje, budou také v následujících kapitolách podrobně popsané základní komponenty, ze kterých se přepínač skládá (například OpenFlow Channel, Group tabulka, Flow tabulka, porty a další). (OpenFlow Switch Specification, 2015, s. 11)

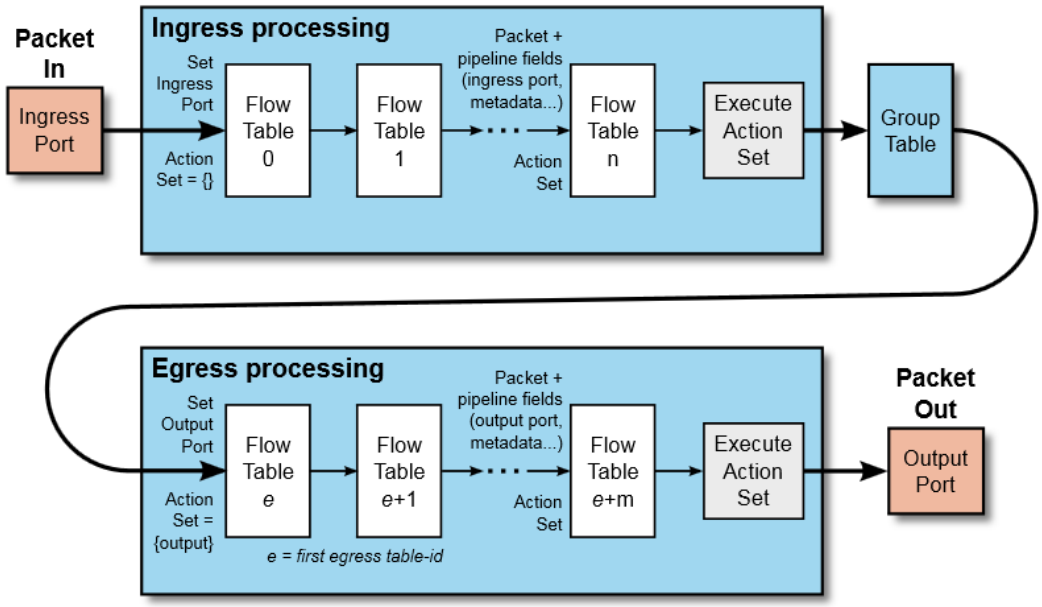
2.1 OpenFlow přepínač

OpenFlow přepínač se skládá z jedné, nebo více Flow tabulek a Group tabulek, které slouží k vyhledávání a přesměrovávání zpráv. Dále obsahuje jeden, nebo více, OpenFlow Channels k externímu kontroléru. OpenFlow přepínač komunikuje s kontrolérem a kontrolér řídí přepínač pomocí OpenFlow protokolu. Zpracování zpráv napříč všemi tabulkami se nazývá pipeline processing V následujících kapitolách je popsané, jak OpenFlow přepínač pracuje. Pomocí OpenFlow Switch protocol může kontrolér přidávat, aktualizovat a mazat Flow pravidla ve Flow tabulkách daného přepínače, a to jak reaktivně (v reakci na OpenFlow zprávu), tak proaktivně. Každá Flow tabulka uvnitř přepínače obsahuje seznam Flow pravidel, kde každé pravidlo se skládá z match fields (porovnávacího kritéria), počítadla a seznamu instrukcí, které se použijí na odpovídající zprávy. (OpenFlow Switch Specification, 2015, s. 12)

[1] Existují i rozdílné implementace, například může veškerou komunikace obstarat kontrolér.

2.2 Zpracování OpenFlow událostí

Proces začíná v první Flow tabulce a může pokračovat do jiné Flow tabulky v dané pipeline[2]. Flow pravidla jsou seřazené podle priority. Pokud tedy příchozí zpráva nesplňuje první pravidlo, tak se přepne na další a tak dále. Při shodě pravidla se zprávou, se provedou instrukce spojené s daným pravidlem. Pokud žádné pravidlo neodpovídá, záleží na konfiguraci takzvaného table-miss Flow entry. Taková zpráva se může například odeslat na kontrolér pomocí OpenFlow channel, nebo může být zahozena, či může být přesměrována do další Flow tabulky. Instrukce spojené s každým Flow pravidlem obsahují různé akce, nebo modifikují proces dané pipeline, ve které se Flow tabulka nachází.



Obrázek 2: OpenFlow pipeline

Zdroj: (*Open Networking Foundation, 2015)*

Flow pravidla mohou přesměrovat zprávu na port. V těchto případech se zpravidla bavíme o fyzických portech, ale mohou být přesměrovány i k logickým portům, definovaných přepínačem, nebo na takzvané rezervované porty. Rezervované porty mohou specifikovat směrovací akce jako je odesílání na kontrolér, flooding nebo přesměrování neužívající OpenFlow metody, tedy „běžný“ přepínací proces.

[2] Pipeline – zpracování postupné, či zřetězené sekvence událostí.

Pipeline proces začíná vždy se zpracováním v první Flow tabulce. Zpráva je porovnáná s Flow pravidlem umístěným ve Flow tabulce s číslem 0. Ostatní tabulky jsou využívány podle výstupu z první Flow tabulky. Když je nalezeno pravidlo, kterému odpovídá příchozí zpráva, provedou se instrukce uvnitř daného Flow pravidla. Zpráva může být poslána do následující Flow tabulky jen v případě, že následující Flow tabulka má větší identifikační číslo než tabulka předchozí. Jinými slovy pipeline se může zpracovávat pouze v jednom směru. Nelze se v pipeline vracet. S tím souvisí, že poslední Flow tabulka nemůže zahrnovat instrukci k zaslání do jiné Flow tabulky. Pokud nějaká Flow tabulka v pipeline neobsahuje instrukci k zaslání do jiné tabulky, je zpravidla zpráva zaslána na výstup z přepínače. (OpenFlow Switch Specification, 2015, s. 19 - 20)

2.3 OpenFlow porty

OpenFlow porty jsou síťové rozhraní určené k přechodu zpráv mezi OpenFlow komponenty a zbytkem sítě. OpenFlow přepínače jsou navzájem spojené pomocí OpenFlow portů. Zpráva může být zaslána z jednoho OpenFlow přepínače do jiného jen přes výstupní OpenFlow port na prvním přepínači do vstupního portu na druhém přepínači.

Počet OpenFlow portů nemusí být identický jako počet který přepínač má hardwarově k dispozici. Některé porty nemusí být ani vyhrazeny pro OpenFlow komunikaci.

Zprávy jsou přijaty na vstupním portu a jsou zpracovány pomocí OpenFlow pipeline, která je může zaslat na výstupní port. Vstupní port je pro příchozí zprávu znám po celou dobu zpracovávání OpenFlow pipeline. Vstupní port může být využit k porovnávání s Flow pravidly. OpenFlow pipeline může rozhodnout na jaký výchozí port má zaslat zprávu pomocí výchozí akce, která definuje, jak má být zpráva zaslána mimo přepínač dále do sítě.

OpenFlow přepínač musí podporovat tři typy OpenFlow portů: fyzické porty, logické porty a rezervované porty. (OpenFlow Switch Specification, 2015, s. 15)

2.3.1 Fyzické porty

OpenFlow fyzické porty jsou přepínačem definované porty, které přímo korespondují s reálnými hardwarovými rozhraními na přepínači. Na běžných ethernetových přepínačích jsou fyzické porty namapovány jedna ku jedné na Ethernetová rozhraní.

V některých nasazeních OpenFlow protokolu, může OpenFlow přepínač být virtualizován na hardwaru přepínače. V takových případech bývají virtualizované OpenFlow fyzické porty namapovány na hardwarové rozhraní přepínače. (OpenFlow Switch Specification, 2015, s. 16)

2.3.2 Logické porty

OpenFlow logické porty jsou přepínačem definované porty, které nekorespondují přímo na hardwarové rozhraní přepínače. Logické porty patří do jisté úrovně abstrakce, které definuje přepínač s využitím pro metody netýkající se OpenFlow (agregační skupiny, tunely, loopback rozhraní a podobné).

Logické porty mohou zapouzdřovat zprávy a mohou být namapovány na různé fyzické porty. Zpracovávání logických portů je implementačně závislé a musí být viditelné pro OpenFlow procesy a musí interagovat s OpenFlow procesy jako OpenFlow fyzický port.

Jediný rozdíl mezi fyzickým a logickým portem je, že zprávy spojené s logickým portem mohou mít extra pipeline pole zvané Tunnel-ID a pokud je zpráva přijata na logickém portu a je zaslána na kontrolér, musí být logický port i fyzický port nahlášeny kontroléru. (OpenFlow Switch Specification, 2015, s. 16)

2.3.3 Rezervované porty

OpenFlow definuje takzvané rezervované porty. Specifikují obecné akce jako zasílání na kontrolér, flooding, nebo zasílání pomocí metod netýkajících se OpenFlow (proces běžného ethernetového switche).

Následně jsou popsány všechny rezervované porty, které switch musí podporovat.

- **ALL:** reprezentuje všechny porty, které switch může použít k odesílání specifických zpráv. Smí být použit pouze jako výstupní port.
- **CONTROLLER:** reprezentuje kontrolní kanál s OpenFlow kontrolérem. Může být použit jako vstupní, i jako výstupní port. Pokud je využit jako výstupní port, zapouzdří

zprávu v packet-in zprávě a zašle ji pomocí OpenFlow switch protokolu. Pokud je využit jako vstupní port, pak identifikuje zprávu, z jakého kontroléru pochází.

- **TABLE:** reprezentuje start OpenFlow pipeline. Tento port je validní pouze ve výstupní akci v seznamu akcí uvnitř packet-out zprávě a předloží zprávu první Flow tabulky, takže zpráva může být zpracována běžnou OpenFlow pipeline.
- **IN_PORT:** reprezentuje vstupní port zprávy. Může být použit jako výstupní port, zasláním zprávy ven skrz jeho vstupní port.
- **ANY:** speciální hodnota využívána v některých OpenFlow požadavcích v případech, kdy není specifikovaný žádný port. Některé OpenFlow požadavky obsahují referenci na specifický port. S využitím ANY jako číslo portu v takových požadavcích umožňují požadavek aplikovat na všechny porty. Může být využit jak pro vstupní, tak pro výstupní porty.
- **UNSET:** speciální hodnota, která specifikuje, že výchozí port nebyl nastaven v Action-set (viz kapitola 2.5 – Action-set). Využívá se jen při pokusu porovnat výchozí port v Action-set s využitím OXM_OF_ACTSET_OUTPUT porovnávacím polem. Může být využit jak pro vstupní, tak pro výstupní porty.

(OpenFlow Switch Specification, 2015, s. 16 -17)

2.3.4 Úpravy portů

OpenFlow protokol může porty na OpenFlow přepínači upravovat dle potřeby. Může je přidávat i odebírat. Kromě přepínače může porty spravovat i kontrolér. Jakákoliv změna provedená přepínačem na portech, musí být zaslána i kontroléru ve formě zprávy o změně stavu přepínače.

Jakákoliv modifikace portů se neprojeví do již uložených Flow pravidel na přepínači. To znamená, že pokud dojde například k vypnutí nějakého portu, na který referuje nějaké Flow pravidlo, budou veškeré zprávy, které toto Flow pravidlo budou splňovat, zahozeny. Stejné pravidla platí i pro Group tabulky.

Čísla smazaných portů mohou být využity pro jiné fyzické, nebo logické porty. Vzhledem k předchozímu pravidlu, lze tímto způsobem dosáhnout efektivního přesměrování na nový port. Nicméně pokud je port smazán a jeho identifikátor není znovu využit, je pak na kontroléru, aby Flow pravidla a případně Group pravidla referující na tento port byly vymazány. (OpenFlow Switch Specification, 2015, s. 17)

2.4 Flow tabulky a Flow pravidla

Flow tabulka je struktura určena k uchovávání, vyhledávání a porovnávání Flow pravidel.
Flow pravidla se skládají z: Match Fields, priority, počítadla, instrukce, Timeouts, Cookies a Flags.

- **Match fields:** pole skládající se z více položek, které se porovnávají s příchozí zprávou[3]. Může být prázdné (jako například u defaultního pravidla), nebo se může skládat z přístupového portu a hlavičky zprávy. Případně z dalších položek, jako metadata, které specifikovala předchozí tabulka.
- **Priorita:** k určení priority pravidla. Čím je priorita vyšší, tím větší má pravidlo „přednost“ před ostatními.
- **Počítadlo:** zvyšuje se s každou shodou zprávy.
- **Instrukce:** modifikuje Action set nebo zpracovávání pipeline.
- **Timeouts:** čas určující, za jak dlouho pravidlo expiruje, či jak dlouho může být nevyužívané.
- **Cookie:** používají se kontrolérem k filtrování Flow pravidel ovlivněné Flow statistikou, Flow modifikací a Flow deletion requests. Nevyužívá se pro porovnávání zpráv.
- **Flags:** upravují, jak je dané Flow pravidlo spravované.[4]

Jako identifikátor záznamu v tabulce se využívá kombinace Match fields a priority. Tyto dvě složky společně tvoří unikátní identifikátor. Flow pravidlo, které nemá vyplněný žádný Match Field a priorita je nastavena na 0 se nazývá Table-miss flow entry.

Instrukce určitého Flow pravidla může obsahovat akce, které se vykonají v určitém bodě v pipeline. (OpenFlow Switch Specification, 2015, s. 22)

2.4.1 Princip porovnávání pravidel

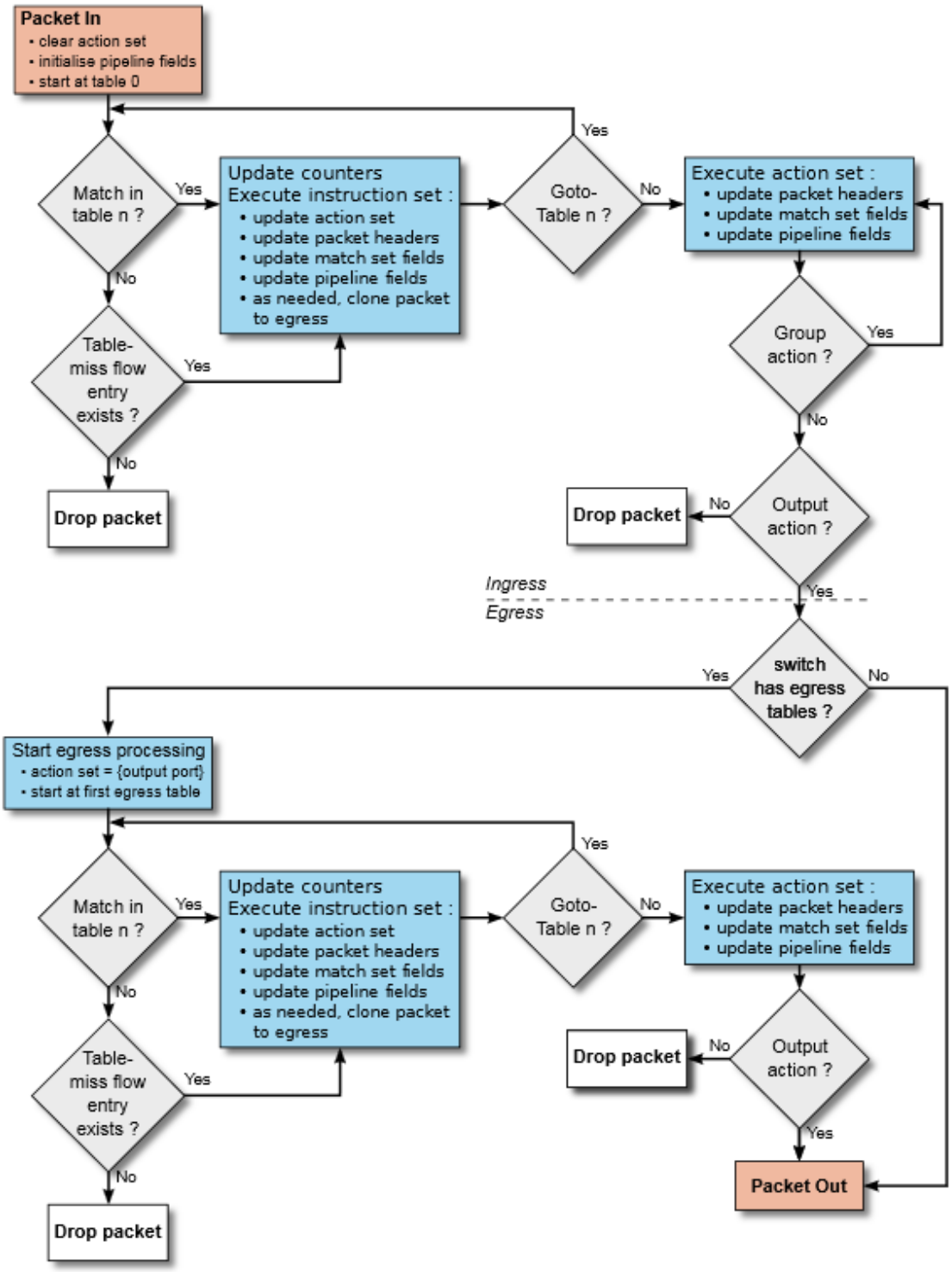
Porovnává se hlavička zprávy. Závisí na typu zprávy a obecně hlavička může obsahovat širokou škálu informací jako ethernetovou zdrojovou adresu, IPv4/IPv6 cílovou adresu, TCP/UDP port a podobně. Porovnání může být provedeno i oproti vstupnímu portu, metadatům a ostatních pipeline polím. Metadata mohou být použita pro odeslání informací mezi tabulkami v přepínači.

[3] Zpráva je obecný název. Konkrétně se může jednat o paket, datagram, či rámec.

[4] Například flag OFPFF_SEND_FLOW_REM definuje, že přepínač bude při odstranění Flow pravidla informovat kontrolér pomocí OpenFlow zprávy.

Hlavička zprávy a pipeline fields reprezentují současný stav, jinými slovy se akce propisují do hlavičky zprávy a nad těmito změnami se může provozovat další porovnání.

Zpráva vyhovuje jen v případě, že dojde ke kompletní shodě. Nelze porovnávat jen na základě částečné shody porovnávacích kritérií. (OpenFlow Switch Specification, 2015, s. 22)



Obrázek 3: Diagram znázorňující zpracování zprávy pomocí OpenFlow přepínače

Zdroj: (*Open Networking Foundation, 2015*)

2.5 Action set

Action set je spojen s každou zprávou. Defaultně je prázdný, ale Flow pravidlo jej může modifikovat pomocí instrukcí Write-Action a Clear-action během každého konkrétního porovnání s Flow pravidlem. Action set se zasílá mezi Flow tabulkami. Pokud Flow pravidlo nezasílá zprávu do jiné tabulky, pak se pipeline proces zastaví a akce uvnitř Action set se nad danou zprávou vykonají. Pokud Action set obsahuje Group akci, tak se vykonají i dané akce uvnitř Action bucket, který je uvnitř Group action.

Action set obsahuje maximálně jednu akci daného typu. Set-field action jsou identifikovány podle pole typu, přičemž Action set obsahuje maximálně jeden z každého set-field action pro každý typ pole. Copy-field action je v Action set nedefinovaná. Experimenter actions jsou identifikovány podle experimenter-id a experimenter-type. Zde opět platí že Action set může obsahovat pouze jeden typ.

Pokud je do Action set přidána akce typu, která již existuje, tak se přepíše za aktuálnější přidávanou verzi.

Action set pro Egress processing má jistá omezení. Do Action set nesmějí být přidány žádné Output action ani Group action. Action set je nastaven na začátku Egress procesu s Output action referující na aktuální output port. Toto je jeden z rozdílů oproti Ingress processing, kde action set je na počátku prázdný.

Při provádění Action set se akce provádějí podle určitého pořadí (dle seznamu, který je uveden níže), bez ohledu na to, v jakém pořadí byly přidány do Action set. Stejné pravidlo platí i pro Action buckets.

- **Copy TTL inwards:** aplikuje kopii TTL pro danou zprávu.
- **Pop:** aplikuje všechny pop akce zprávy.
- **Push-MPLS:** aplikuje MPLS tag push akce zprávy.
- **Push-PBB:** aplikuje PBB tag push akce zprávy.
- **Push-VLAN:** aplikuje VLAN tag push akce zprávy.
- **Copy TTL outwards:** aplikuje kopii TLL vnější akce zprávy.
- **Decrement TTL:** aplikuje dekrementační TTL akce zprávy.
- **Set:** aplikuje všechny set-field akce zprávy.
- **QoS:** aplikuje všechny QoS akce, jako meter a set_queue zprávy.
- **Group:** pokud existuje group akce, aplikuje akce příslušného group bucket v pořadí, které specifikuje daný bucket.
- **Output:** pokud neexistuje group akce, zašle zprávu na port specifikovaný output action.

V případě že existuje Group action je Output action ignorován a provede se pouze Group action. Pokud neexistuje action ani jednoho typu je zpráva zahozena. Existuje alternativa, že Group action a Output action existuje, ale Output action referuje na neexistující port. V takovém případě je zpráva také zahozena.

Output action se zpracovává rozdílně v Ingress procesu a Egress procesu. Když Ingress action set obsahuje Output action, nebo Group action, který zasílá zprávu na port, pak musí začít Egress proces na stejném portu. To samé platí i v případě, že Output action referuje na všechny rezervované porty. To znamená, že Egress proces musí začít na každém portu zvlášť, kam byla kopie zprávy zaslána. Pokud Egress action set obsahuje Output action, zpráva musí opustit Egress proces a musí být zpracován portem. Nejčastěji je odeslán ven z přepínače. (OpenFlow Switch Specification, 2015, s. 26)

2.6 Group Table

Další akce spojená s Flow pravidly je přímé přesměrování do Group, které specifikuje další možnosti, jak s pracovat se zprávou. Group reprezentují seznam akcí pro flooding a více komplexní směrovací techniky jako multipath, fast reroute, link aggregation.

Group table se skládá z Group pravidel. Možnost přesměrovat zprávu pomocí běžných Flow pravidel do „Group“ umožňuje OpenFlow využít další metody co s danou zprávou provést.

Hlavička tabulky se skládá z Group Identifier, Group Type, Counters a Action Buckets.

- **Group Identifier**: 32-bitové unikátní, nezáporné celé číslo identifikující danou Groupu v OpenFlow přepínači.
- **Group Type**: určuje Group sémantiku. Mezi základní typy patří indirect a all.[5]
- **Counters**: mění se, když je zpráva zpracována danou Groupou.
- **Action Buckets**: seřazený seznam určitých „Buckets“ (balíků) akcí, kde každý balík obsahuje set akcí, které se mají provést a parametry.

Group pravidlo nemusí obsahovat žádný, jeden, nebo více Action Buckets. Action Bucket obvykle obsahuje akce, které modifikují zprávu a následně jej pomocí další akce zašle na výstup na konkrétní port. Bucket může také obsahovat Group akci, která invokuje další Groupu (pouze

[5] Indirect – provede 1 definovaný bucket v této Group.
All – provede všechny bucket v této Group.

v případě, že přepínač podporuje Group chaining). V takovém případě je zpráva zpracovávána dalšími akcemi. (OpenFlow Switch Specification, 2015, s. 33)

2.7 Ingress a Egress processing

Flow tabulky mohou být použity jak pro Ingress processing, tak pro Egress processing. Ingress processing se provádí vždy, když zpráva přijde do OpenFlow přepínače a může zahrnovat jednu nebo více Flow tabulek. Egress processing se provádí až po určení output portu a nemusí na rozdíl od Ingress processing zahrnovat žádnou Flow tabulku.

Je zde několik rozdílů mezi používáním Flow tabulek v Ingress a Engress processingu. Na počátku Ingress processing je Action set prázdný. Flow tabulky mohou zasílat zprávu do jiných Ingress Flow tabulek pouze pomocí instrukce Goto-Table. Pomocí této instrukce nelze zasílat zprávu z Ingress Flow tabulky do Egress Flow tabulky.

Na začátku Egress processing, Action set dané zprávy, obsahuje pouze Output akci pro konkrétní output port. Pro output akci jsou validní jakékoliv fyzické, čí logické porty, nebo CONTROLLLER, či LOCAL rezervované porty. Egress Flow tabulky mohou zasílat zprávy pouze pomocí instrukce Goto-Table do jiných Egress Flow tabulek. Dále musí podporovat „Action Set Output match field“. Je to z důvodu aby Flows vycházely z kontextu pro output port. Jako další musí být zakázáno použití output akce, nebo group akce ve write-action instrukci, aby nebylo možné měnit output port. Tato omezení musí být dodrženy ve funkcích Flow tabulek.

Egress table mohou volitelně podporovat Output akci, nebo Group akci v apply-action instrukci. Tyto akce se chovají jako v Ingress tabulkách, zasílají kopii zprávy na specifikovaný port, kde se spustí další Egress proces. Toto může být použito například pro zrcadlení komunikace. Přepínač by ale měl být ošetřen tak, aby nedošlo k nekonečným smyčkám, kdy Group akce a Output akce mohou donekonečna zasílat kopie zpráv stále na stejnou tabulku v Egress procesu. Takové ošetření však spadá mimo specifikaci protokolu OpenFlow. (OpenFlow Switch Specification, 2015, s. 37)

2.8 OpenFlow Channel a Control Channel

OpenFlow Channel je rozhraní, které spojuje každý logický OpenFlow přepínač s OpenFlow kontrolérem. Skrz toto rozhraní kontrolér konfiguruje a spravuje přepínač, získává zprávy z přepínače a zasílá zpráv ven z přepínače.

Control Channel může podporovat jeden OpenFlow Channel s jedním kontrolérem, nebo několik OpenFlow channelů povolující několik kontrolérů ke sdílení správy přepínače.

Mezi přepínačem a OpenFlow Channel lze implementovat rozhraní více způsoby, nicméně všechny OpenFlow channely zprávy musí být ve formátu, který podporuje OpenFlow Switch protokol. Obecně se OpenFlow Channel šifruje s využitím TLS, ale může běžet i přímo přes TCP.[6] (OpenFlow Switch Specification, 2015, s. 38)

2.9 Typy OpenFlow zpráv

OpenFlow přepínač podporuje tři typy zpráv: controller-to-switch, asynchrounous a symmetric. Každý má několik sub-typů. Controller-to-switch zprávy jsou vytvářeny kontrolérem a využívají se přímo ke správě, nebo kontrole stavu přepínače. Asynchronous zprávy jsou vytvářeny přepínačem a využívají se k aktualizaci kontroléru o síťových událostech a změn ve stavu přepínače. Symmetric zprávy mohou vytvářet jak kontrolér, tak přepínač, a jsou zasílány bez jakékoliv žádosti. Využívají se k navazování nových spojení, ke kontrole spojení mezi kontrolérem a přepínačem nebo k obeznámení kontroléru o chybě. (OpenFlow Switch Specification, 2015, s. 38)

2.9.1 Controller-to-switch

Controller-to-switch jsou vytvářeny kontrolérem a mohou požadovat od přepínače odpověď. Níže jsou popsány typy zpráv, které může kontrolér vytvářet.

- **Features**: kontrolér může požadovat po přepínači svůj identifikátor a základní přehled funkcionalit, které přepínač poskytuje. Přepínač na tento typ zprávy musí odpovědět. Běžně se tato zpráva využívá po vytvoření OpenFlow Channel.
- **Configuration**: kontrolér je schopný nastavit a získat konfiguraci daného přepínače. Přepínač odpovídá jen v případě, že se jedná o dotaz, nikoli o nastavení.
- **Modify-State**: využívá se ke správě stavu na přepínači. Primární účel je k přidání, smazání, a modifikovaní Flow/Group pravidel a vkládání, čí odebírání Action buckets z Group. Dále se využívá k nastavení vlastností jednotlivých portů.

[6] Defaultně je OpenFlow Channel nešifrován.

- **Read-State**: kontrolér jej využívá k získávání informací jako: aktuální konfigurace, statistické údaje a možné funkcionality. Většina těchto zpráv je implementována s využitím vícečetných sekvencí (ne pouze pomocí jedné velké zprávy, limit jedné zprávy je pouze 64 KB).
- **Packet-out**: jsou využívány kontrolérem k zaslání zprávy ven ze specifikovaného portu na přepínači[7] a k zaslání zpráv získané přes Packet-in zprávu. Packet-out zpráva musí obsahovat celou zprávu, nebo buffer ID referující na zprávu uloženou v přepínači. Zpráva musí také obsahovat seznam akcí v pořadí ve kterém bylo původně specifikováno. Pokud je seznam prázdný je zpráva zahozena.
- **Barrier**: zpráva, kterou kontrolér využívá k ověření závislostí, nebo k získání odpovědí o dokončené operaci.
- **Role-request**: používá se k nastavení role OpenFlow Channelu a nastavení Controller ID. Kromě nastavení se tato zpráva využívá i k získání těchto informací. Nejčastěji se využívá, když přepínač je připojen k několika kontrolérům současně.
- **Asynchronous-Configuration**: používá se kontrolérem k nastavení, či získání, filtrace asynchronous zpráv, které chce získat z OpenFlow Channelu.

(OpenFlow Switch Specification, 2015, s. 38)

2.9.2 Asynchronous

Asynchronous zprávy jsou odesílány, aniž by o ně kontrolér žádal přepínač. Přepínače jej využívají k obeznámení, že přišla zpráva, nebo že přepínač upravil svůj stav. Hlavní typy zpráv jsou popsané níže.

- **Packet-in**: přepínač předá kontrolu nad zprávou kontroléru. Kdykoliv se přesměruje zpráva na kontrolér s využitím Flow pravidla, nebo instrukce table-miss flow entry, pak je vždy využit packet-in. Jiné procesy, jako TTL ověření, mohou také vygenerovat packet-in a zaslat zprávu na kontrolér.

[7] Může být i zaslán do jiné tabulky, nebo být zahozen.

- **Flow-Removed**: informuje kontrolér o odebrání Flow pravidla z Flow tabulky. Flow-Removed zprávy jsou zasílány pouze pro Flow pravidla s příznakem OFPFF_SEND_FLOW_REM. Tyto zprávy jsou generované jako reakce na požadavek kontroléru o smazání Flow pravidla, nebo pokud nějakému Flow pravidlu přeteče čas, po kterou může pracovat (timeout).
- **Port-status**: informuje kontrolér o změně portu. Očekává se, že přepínač zašle Port-status kontroléru jako konfiguraci portů, nebo změnu stavu portu. Tyto události zahrnují i změny v konfiguraci portů, například pokud port byl deaktivován uživatelem.
- **Role-status**: informuje kontrolér o změně role. Když nový kontrolér zvolí sebe za mastera[8], pak se očekává že přepínač zašle role-status zprávu bývalému master kontroléru.
- **Controller-Status**: informuje kontrolér o změnách stavu OpenFlow Channelu. Přepínač zasílá tyto zprávy všem kontrolérům, pokud se stav OpenFlow Channelu změní jakémukoliv přepínači. Tato zpráva může pomoci, když kontroléry ztratí možnost komunikovat mezi sebou.
- **Flow-monitor**: informuje kontrolér o změnách ve Flow tabulkách. Kontrolér může definovat seznam monitorů ke sledování těchto změn.

(OpenFlow Switch Specification, 2015, s. 39)

2.9.3 Symmetric

Symmetric zprávy jsou zasílány obousměrně, bez jakéhokoliv požadavku.

- **Hello**: hello zprávy jsou zasílány mezi přepínačem a kontrolérem během navazování spojení.
- **Echo**: echo požadavek/odpověď zprávy mohou být zaslány obousměrně. Na echo požadavek musí být vždy vytvořena odpověď. Využívají se hlavně k ověření spojení mezi přepínačem a kontrolérem a mohou být využity k měření latence nebo šířce pásma.

[8] Kontrolér má jednu z následujících rolí: Equal (defaultní role, má plný přístup k přepínači), Slave (má možnost pouze číst stav přepínače), Master (má plný přístup k přepínači, ale pouze jeden kontrolér může mít tuto roli, ostatní musí mít roli Slave).

- **Error**: error zprávy jsou využity k obeznámení s problémem na obou stranách spojení. Nejvíce jej využívá přepínač k obeznámení kontroléru o neúspěchu nějaké akce, kterou kontrolér požadoval.
- **Experimenter**: využívají se pro doplňkové služby. Jedná se o oblast, která se využívá pro budoucí revize OpenFlow.

(OpenFlow Switch Specification, 2015, s. 40)

2.10 OpenFlow Channel – spojení

Většinou kontrolér spravuje několik OpenFlow Channelů, každý do jiného OpenFlow přepínače. OpenFlow přepínač může mít jeden channel do jednoho kontroléru, nebo i několik channelů, každý do jiného kontroléru. To se využívá zejména pro zajištění spolehlivosti, kdyby nějaký kontrolér, či linka vypadly. Jediný požadavek k navázání spojení je podpora TCP/IP připojení.

OpenFlow Channel je obvykle implementován jako jedno síťové připojení mezi přepínačem a kontrolérem s využitím TLS nebo prostým TCP. Je zde možnost OpenFlow Channel připojit i s využitím několika fyzických linek k využití paralelismu. OpenFlow přepínač zpravidla vytváří OpenFlow Channel, ale je možné tuto funkcionalitu nechat na kontroléru. Musí se ale zajistit zvláštní bezpečnostní opatření (secured connection), aby se předešlo neautorizovaným připojením. (OpenFlow Switch Specification, 2015, s. 41)

2.10.1 Connection URI

Přepínač identifikuje spojení s kontrolérem pomocí unikátního Connection URI. Tato URI musí vyhovovat syntaxi definované v RFC 3986, zejména s ohledem na kódování znaků. Connection URI může mít jeden z následujících tvarů:

- protocol:name-or-address:port,
- protocol:name-or-address.

Protocol zajišťuje přemístění OpenFlow zpráv. Hodnoty pro protocol jsou tls nebo tcp.

Name-or-address je hostname nebo IP adresa kontroléru. Pokud je hostname nakonfigurován jako lokální, pak se doporučuje, aby URI využívalo IP adresu. V případě využití IPv6 by se adresa měla uzavírat v hranatých závorkách, jak je doporučeno v RFC 2732.

Port představuje transportní port využívaný kontrolérem. Pokud není vyplněný, pak se předpokládá využití defaultního portu OpenFlow. Defaultní port OpenFlow pro verzi 1.3 je 6633 (od verze 1.4 je využíván port 6653). (OpenFlow Switch Specification, 2015, s. 42)

2.10.2 Navázání spojení

Přepínač musí mít možnost navázat komunikaci s kontrolérem pomocí Connection URI a transportním portem, který je defaultně nastaven na 6653. Pokud je Connection URI na přepínači správně nakonfigurované, pak přepínač zavede spojení standardně pomocí TLS nebo TCP na kontrolér. Pro správné out-of-band spojení musí přepínač zajistit, aby komunikace do a z OpenFlow Channelu nezpracovávala OpenFlow pipeline. Pro in-band spojení přepínač musí nastavit správný seznam Flow pravidel pro připojení do OpenFlow Pipeline.

Lze využít i možnost, že přepínač nechá na kontroléru, aby navázal spojení. V takovém případě by přepínač měl akceptovat příchozí spojení (standardně TLS nebo TCP) z kontroléru s využitím transportního portu. V obou případech, kdy spojení naváže přepínač, nebo kontrolér, jakmile je spojení navázáno, je poté chování spojení stejné.

Když je připojení poprvé iniciované, musí se z obou stran okamžitě zaslat OFPT_HELLO zpráva s uvedenou nejvyšší možnou verzí, kterou odesílatel podporuje. Hello zpráva může obsahovat i elementy které usnadní nastavení spojení. Po obdržení této zprávy musí příjemce určit verzi OpenFlow Switch protokolu, který má být využit.

Po úspěšném nastavení verze může spojení pokračovat. V jiném případě příjemce musí odeslat Hello Failed chybovou zprávu a spojení musí být ukončené.

Po úspěšné výměně OFPT_HELLO zpráv mezi kontrolérem a přepínačem a úspěšném nastavení verze je základní navázání spojení dokončené a standardní OpenFlow zprávy mohou být zasílané přes toto spojení. Jako jedna z prvních zpráv by měla být zaslána od kontroléru, aby získal Datapath ID daného přepínače. Tato zpráva se nazývá OFPT_FEATURES_REQUEST. (OpenFlow Switch Specification, 2015, s. 43)

2.10.3 Šifrování komunikace

Základním bezpečnostním mechanismem OpenFlow protokolu je Transport Layer Security (TLS). Přepínač a kontrolér mohou komunikovat pomocí TLS spojení k zajištění autentizace a

šifrované komunikace. Přepínač a kontrolér by měl využívat zabezpečenou verzi TLS. Doporučuje se využívat verzi 1.2 či vyšší.

Využití TLS je specifikováno v Connection URI a využívá se buď port specifikovaný uživatelem, nebo defaultní port 6653. Spojení může iniciovat jak kontrolér, tak i přepínač. Běžně se však využívá, že komunikaci zakládá přepínač.

Přepínač a kontrolér se vzájemně ověřují výměnou certifikátů podepsaných specifickým soukromým klíčem. Jedná se o běžný a doporučený způsob zabezpečené komunikace. Každý přepínač musí být uživatelem nakonfigurovaný s privátním a veřejným certifikátem, které jsou využity pro ověření kontroléru. Přepínač může navíc podporovat konfiguraci několika CA certifikátů a udržuje Certificate Revocation Lists (CRLs) při navazování spojení s několika kontroléry. Je doporučené nakonfigurovat a spravovat všechny související bezpečnostní pověření užitím Switch Management protokolu jako je OpenFlow Configuration protokol.

V případě využití TCP nešifrovaného připojení, je doporučené využít alternativního zabezpečeného spojení (například IPsec, VPN, separované fyzické připojení atd.). (OpenFlow Switch Specification, 2015, s. 46)

2.10.4 Přerušení komunikace

V případě ztráty komunikace mezi přepínačem a kontrolérem, musí přepínač zaslat Controller Status zprávu do všech zbývajících kontrolérů (pokud existují). Controller Status zpráva obsahuje roli a stav control channelu problémového kontroléru. V případě, že přepínač ztratí komunikaci s více kontroléry, musí zaslat Control Status zprávu všem zbývající kontrolérům (pokud existují) a to za všechny kontroléry se kterými daný přepínač ztratil komunikaci. To umožňuje ostatním kontrolérům jednat i v případě, že mezi sebou nemají žádné spojení. Pokud se OpenFlow Channel obnoví, pak přepínač musí o této události uvědomit všechny zbývající kontroléry pomocí Controller Status zprávy.

V případě, že přepínač ztratí kontakt se všemi kontroléry (například z důvodu vypršení TLS session, nebo fyzického odpojení), musí přepínač okamžitě přejít do režimu „fail secure mode“ nebo „fail standalone mode“. Záleží na konfiguraci přepínače, do jakého režimu přejde. Když se přepínač nachází v režimu „fail secure mode“, tak jediná změna v chování přepínače je, že zprávy a OpenFlow zprávy které jsou určeny pro kontrolér jsou zahozené. Flow pravidla by měly expirovat podle časového limitu určeného uvnitř pravidla (Timeout).V „fail standalone mode“ přepínač zpracovává zprávy s využitím OFPP_NORMAL rezervovaného portu. Jinými

slovy se přepínač chová jako „starší“ Ethernetový přepínač, nebo směrovač. V režimu „fail standalone mode“ může přepínač volně využívat Flow tabulky podle potřeby. Může jednotlivé Flow pravidla mazat, přidávat a modifikovat. Fail standalone mode je k dispozici převážně na hybridních přepínačích[9].

Po obnovení OpenFlow Channelu jsou veškeré OpenFlow operace obnoveny. Kontrolér může být nastaven, aby po obnovení komunikace s přepínačem, si zažádal o všechna Flow pravidla pomocí flow-stats žádosti a resynchronizoval svá Flow pravidla s Flow pravidly přepínače. Další možnost je, že kontrolér pomocí flow-mod žádosti smaže veškeré Flow pravidla z přepínače a začne je nastavovat na čistém přepínači od začátku.

Přepínač se nachází v režimu „fail secure mode“, nebo „fail standalone mode“ již při prvním startu a operuje v něm, dokud nedojde k prvnímu navázání s kontrolérem pomocí OpenFlow Channel. (OpenFlow Switch Specification, 2015, s. 45)

2.11 Modifikace Flow tabulky

Zprávy s jejichž pomocí lze modifikovat Flow tabulky mohou být následujících typů:

```
enum ofp_flow_mod_command {

OFPFC_ADD           = 0, /* Nové pravidlo. */

OFPFC_MODIFY        = 1, /* Modifikuje všechny odpovídající Flow
pravidla. */

OFPFC_MODIFY_STRICT = 2, /* Modifikuje pravidlo, které striktně od-
povídá masce a prioritě */

OFPFC_DELETE        = 3, /* Smaže všechny odpovídají pravidla. */

OFPFC_DELETE_STRICT = 4, /* Smaže pravidlo, které striktně odpovídá
masce a prioritě. */

};
```

Při Add požadavku pro přidání musí přepínač nejdříve zkontrolovat, zda nedochází k překrývání Flow pravidel. Dvě Flow pravidla se překrývají, pokud se zprávy s oběma shoduje a pokud

[9] Hybridní přepínač – kombinuje zpracování pomocí OpenFlow pipeline a režim klasického přepínaní. Lze například pomocí VLAN tagu rozlišovat, kterým způsobem zpracovat příchozí zprávu. Mimo jiné je možné, aby zpráva byla předána z OpenFlow do režimu klasického přepínaní.

obě Flow pravidla mají stejnou prioritu, ale zároveň nemají tyto pravidla naprosto totožná pravidla pro porovnání se zprávou. Pokud k takovému konfliktu dojde, pak přepínač musí zamítnout přidání takového Flow pravidla a zašle Overlap error zprávu jako odpověď kontroléru.

V případě že k překrývání nedochází, přepínač musí vložit Flow pravidlo do požadované tabulky. Pokud Flow pravidlo s identickým porovnávacím pravidlem a prioritou již existuje ve Flow tabulce, pak takové pravidlo musí být vymazané z Flow tabulky a musí být nahrazené novým Flow pravidlem. Počítadlo daného Flow pravidla může být vynulováno, nebo přejato od předchozího Flow pravidla (záleží na nastavení příznaku OFPFF_RESET_COUNTS). V případě nahrazení Flow pravidla není generována žádná zpráva pro kontrolér o této události.

Modify požadavek (platí pro OFPFC_MODIFY i pro OFPFC_MODIFY_STRICT) upraví dané pravidlo (pokud existuje). Úpravy probíhají tak, že se přepíší instrukce daného Flow pravidla z Modify požadavku, který zasílá kontrolér. Cookie, idle_timeout, hard_timeout, importance a flags jsou nezměněny. Stejně jako u Add požadavku, i zde může být počítadlo vynulováno pomocí OFPFF_RESET_COUNTS. Pokud Flow tabulka neobsahuje žádné pravidlo, které by se shodovalo s požadavkem, pak není žádné pravidlo upravené a není vyvolána žádná chyba.

Delete požadavek (platí jak pro OFPFC_DELETE, tak i pro OFPFC_DELETE_STRICT), v případě že takové Flow pravidlo existuje, smaže jej. Pokud pravidlo má nastavené OFPFF_SEND_FLOW_REM, pak musí být vygenerována Flow removed zpráva. Pokud neexistuje Flow pravidlo, které se shoduje s Delete požadavkem, pak není vyvolána žádná chyba.

Strict verze u Delete a Modify určují, zda mají být Flow pravidla porovnávány striktně. Jinými slovy u Strict verze se daný požadavek provede jen v případě, že Flow pravidla jsou ve všem identické s požadavkem, zatímco u Non-Strict verze může stačit jen částečná shoda. (OpenFlow Switch Specification, 2015, s. 51)

2.12 OpenFlow Switch Protocol

Nyní, když byly vysvětleny veškeré základní komponenty OpenFlow protokolu a základní principy komunikace OpenFlow protokolu, je v následujících kapitolách popsán OpenFlow jako celek. OpenFlow protokol je implementován s využitím OpenFlow zpráv zasílaných přes OpenFlow Channel. Každý typ zprávy má specifickou strukturu, která začíná klasicky OpenFlow hlavičkou. Každá struktura definuje pořadí, ve kterém jsou informace zahrnuty ve zprávě a může obsahovat i jiné struktury, hodnoty, výčtové typy nebo bitmask.

Struktury, definice a výčty popsané níže jsou oddděné z hlavičkového souboru openflow.h.

OpenFlow je v podstatě jednoduchý binární protokol. Z tohoto důvodu chybné OpenFlow zprávy generované nějakým OpenFlow zařízením mohou ve výsledku podávat zprávy s chybnými hodnotami. Proto je doporučeno monitorovat tyto situace a detekovat nekompilovatelné zprávy. (OpenFlow Switch Specification, 2015, s. 62)

2.13 OpenFlow Header

Každá OpenFlow zpráva začíná s OpenFlow hlavičkou:

```
/* Hlavička OpenFlow paketu. */
struct ofp_header {
uint8_t version;      /* OFP_VERSION. */
uint8_t type;         /* Jedna z OFPT_ constants. */
uint16_t length;      /* Délka této hlavičky. */
uint32_t xid;         /* ID transakce přidružené k tomuto paketu.*/
};
OFP_ASSERT(sizeof(struct ofp_header) == 8);
```

Version popisuje verzi OpenFlow Switch protokolu. První bit je rezervovaný a musí být nastaven vždy na 0. Ostatních 7 bitů indikuje číslo revize protokolu.

Length udává velikost zprávy.

Type může mít následující hodnoty:

```
enum ofp_type {
/* Immutable messages. */
OFPT_HELLO              = 0,  /* Symmetric message */
OFPT_ERROR              = 1,  /* Symmetric message */
OFPT_ECHO_REQUEST       = 2,  /* Symmetric message */
OFPT_ECHO_REPLY         = 3,  /* Symmetric message */
OFPT_EXPERIMENTER       = 4,  /* Symmetric message */
/* Switch configuration messages. */
OFPT_FEATURES_REQUEST   = 5,  /* Controller/switch message */
```

```
OFPT_FEATURES_REPLY     = 6,  /* Controller/switch message */
OFPT_GET_CONFIG_REQUEST = 7,  /* Controller/switch message */
OFPT_GET_CONFIG_REPLY   = 8,  /* Controller/switch message */
OFPT_SET_CONFIG         = 9,  /* Controller/switch message */

/* Asynchronous messages. */
OFPT_PACKET_IN          = 10, /* Async message */
OFPT_FLOW_REMOVED       = 11, /* Async message */
OFPT_PORT_STATUS        = 12, /* Async message */

/* Controller command messages. */
OFPT_PACKET_OUT         = 13, /* Controller/switch message */
OFPT_FLOW_MOD           = 14, /* Controller/switch message */
OFPT_GROUP_MOD          = 15, /* Controller/switch message */
OFPT_PORT_MOD           = 16, /* Controller/switch message */
OFPT_TABLE_MOD          = 17, /* Controller/switch message */
/* Multipart messages. */
OFPT_MULTIPART_REQUEST      = 18, /* Controller/switch message */
OFPT_MULTIPART_REPLY        = 19, /* Controller/switch message */
/* Barrier messages. */
OFPT_BARRIER_REQUEST    = 20, /* Controller/switch message */
OFPT_BARRIER_REPLY      = 21, /* Controller/switch message */
/* Controller role change request messages. */
OFPT_ROLE_REQUEST       = 24, /* Controller/switch message */
OFPT_ROLE_REPLY         = 25, /* Controller/switch message */
/* Asynchronous message configuration. */
OFPT_GET_ASYNC_REQUEST  = 26, /* Controller/switch message */
OFPT_GET_ASYNC_REPLY    = 27, /* Controller/switch message */
OFPT_SET_ASYNC          = 28, /* Controller/switch message */
```

```
/* Meters and rate limiters configuration messages. */
OFPT_METER_MOD          = 29, /* Controller/switch message */
/* Controller role change event messages. */
OFPT_ROLE_STATUS        = 30, /* Async message */
/* Asynchronous messages. */
OFPT_TABLE_STATUS       = 31, /* Async message */
/* Request forwarding by the switch. */
OFPT_REQUESTFORWARD     = 32, /* Async message */
/* Bundle operations (multiple messages as a single operation). */
OFPT_BUNDLE_CONTROL     = 33, /* Controller/switch message*/
OFPT_BUNDLE_ADD_MESSAGE = 34, /* Controller/switch message */
/* Controller Status async message. */
OFPT_CONTROLLER_STATUS  = 35, /* Async message */
};
```

(OpenFlow Switch Specification, 2015, s. 64)

2.13.1 Struktura portů

OpenFlow pipeline zasílá a přijímá zprávy na portech. Přepínač může definovat fyzické a logické porty a protokol OpenFlow může specifikovat i rezervované porty.

Každý port na přepínači má unikátní identifikátor označen jako číslo portu. Jedná se o 32 bitové číslo. Rezervované porty mají identifikační čísla definované protokolem OpenFlow. Čísla logických a fyzických portů se pohybují v rozsahu od 1 do OFPP_MAX. Čísla portů používají následující konvenci:

```
* Číslování portů. Začínají na čísle 1. */

enum ofp_port_no {

/* Maximální počet fyzických a logických portů. */

OFPP_MAX  = 0xffffff00,

/* Rezervovaný OpenFlow Port (fake output "ports"). */

OFPP_UNSET = 0xfffffff7,  /* Output port nenastaven uvnitř action-
set. Využívá se pouze v OXM_OF_ACTSET_OUT_PUT. */

OFPP_IN_PORT = 0xfffffff8,  /*Zašle paket ze vstupního portu. Tento
rezervovaný port musí být explicitně využit k odeslá  ní zpět ze
vstupního portu*/

OFPP_TABLE = 0xfffffff9,  /*Zašle paket do první Flow tabulky. Tento
port lze využít pouze v packet-out zprávách */

OFPP_NORMAL = 0xfffffffa,  /* Zpracování paketu pomocí běžného pře-
píná  ní. */

OFPP_FLOOD  = 0xfffffffb,  /* Flood využívající běžné přepíná  ní.
*/

OFPP_ALL    = 0xfffffffc,  /* Všechny standard  ní porty, kromě
vstupního portu. */

OFPP_CONTROLLER = 0xfffffffd,  /* Zašli kontroléru. */

OFPP_LOCAL  = 0xfffffffe,  /* Lokální openflow "port". */

OFPP_ANY  = 0xffffffff   /* Speciální hodnota použitá v některých
požadavcích, když není zadán žádný port. */

};
```

(OpenFlow Switch Specification, 2015, s. 66)

3 ZAŘÍZENÍ A NÁSTROJE K PRAKTICKÉ ČÁSTI

Tato kapitola popisuje nástroje použité v praktické části této diplomové práce. Je zde popsán přepínač od společnosti HPE, konkrétně model Aruba 3810M. Je zde také popsána implementace protokolu OpenFlow verze 1.3 a testovací nástroje iPerf, směrovač MikroTik a protokol ICMP.

3.1 Kontrolér Ryu

Ryu je komponentově založený framework pro softwarově definované sítě. Je kompletně napsán v jazyce Python. Kromě protokolu OpenFlow podporuje celou řadu dalších funkcionalit jako: Netconf, OF-config a podobné. Veškeré zdrojové kódy jsou volně ke stažení pod licencí Apache 2.0 license. (Build SDN Agilely, 2017)

V této práci je využit SDN Hub Tutorial image, pro VirtualBox, který obsahuje operační systém Linux Ubuntu 14.04 64-bit s připravenou instancí Ryu kontroléru.

3.1.1 Ryu – struktura

Hlavní skripty jsou organizované ve složce /ryu. V případě systému Linux Ubuntu verze 3.13.0-24, který je využit v této práci, se Ryu nachází v: /home/ubuntu/ryu/ryu/. Níže jsou popsány základní komponenty, se kterými Ryu pracuje.

- **app/**: obsahuje množinu aplikací, které poskytují základní síťové funkcionality.
- **base/**: obsahuje třídu pro Ryu aplikaci. RyuApp třída v app_manager.py souboru a je oddděná při vytváření nové aplikace.
- **controller/**: obsahuje množinu souborů, potřebnou ke správné funkcionalitě OpenFlow protokolu. (generování flows, sestavování statistik, zasílání zpráv atd.).
- **lib/**: obsahuje knihovny určené k parsování různých hlaviček protokolů.
- **ofproto/**: obsahuje specifické informace a parsery pro podporu různých verzí OpenFlow protokolů.
- **topology/**: obsahuje kód, který provádí zjišťování topologie náležící OpenFlow přepínači a udržuje s tím spojené informace jako porty, linky atd. Využívá protokol LLDP.

(Ryu Controller Tutorial, [b. r.])

3.2 Aruba 3810M – specifikace

Aruba 3810M přepínač poskytuje výkon a odolnost, který je určený hlavně pro malé a středně velké firmy. Poskytuje širokou škálu funkcionalit, které lze využít například při administraci, či troubleshootingu[10]. Pro zaměření této práce je stěžejní, že se jedná o Layer 3 switch[11] s podporou OpenFlow protokolu.

3.2.1 Technická specifikace

- **Vstupní napětí:** 240 V – může se lišit v různých zemí.
- **Externí I/O porty:** 48 portů 16 1/10GbE SFP+, podporují MACSec 8 HPE Smart Rate Multi-Gigabit (1/2.5/5/10GBASE-T) porty.
- **Latence:** 1000 Mb Latence: < 2,8 µs, 10 Gbps Latence: < 1,8 µs, 40 Gbps Latence: < 1,5 µs.
- **Virtualizace: podpora až** 10 virtuálních switchů
- **Paměť a procesor:** Dual ARM Coretex A9 @ 1 GHz 2 GB DDR3 SDRAM Packet buffer size: 13.5 MB Internal
- **PoE:** maximálně 1440 W
- **Správa funkcionalit:** Aruba AirWave Network Management Aruba Central IMC - Intelligent Management Center Command-line, rozhraní přes webový prohlížeč.

(Aruba 3810M Switch Series - Specifications, [b. r.])

3.2.2 Aruba základní konfigurace

Tato podkapitola je zaměřená na základní konfiguraci přepínače Aruba. Pro připojení k zařízení je využit konzolový port, který je na Aruba přepínači vyhrazen. Pro připojení k příkazové řádce je využit software PuTTY.

Pro úspěšném připojení se otevře příkazová řádka s názvem přepínače.

Konkrétně se tedy zobrazí:

[10] Troubleshooting – proces hledání a řešení chyb

[11] L3 switch – zajišťuje komunikaci mezi ví ce podsítěmi, umí směrovat

```
Aruba-3810M-48G-PoEP-1-slot>
```

V tomto stavu je již možné psát příkazy. Strukturou a syntaxí se velmi podobá příkazové řádce jakou má operační systém IOS od společnosti CISCO. I zde jsou různé úrovně konfigurace a zabezpečení jako je: enable, config a jejich možnost zaheslovaní. To znamená, že na každou úroveň lze stanovit jiné heslo pro přístup.

Dále Aruba poskytuje možnost standardního připojení pro vzdálenou správu pomocí Telnetu, či šifrované možnosti pomocí SSH.

Následuje malá ukázka, jak upravit message of the day:

```
Aruba-3810M-48G-PoEP-1-slot> enable

Aruba-3810M-48G-PoEP-1-slot# configure

Aruba-3810M-48G-PoEP-1-slot(config)# banner motd *

Enter TEXT message.  End with the character '*'

Vitej u sve diplomky*

Aruba-3810M-48G-PoEP-1-slot(config)# exit
```

Message of the day je textová zpráva, která se zobrazí uživateli vždy, když se připojí k zařízení. Lze využít například pro vypsání odpovědné osoby, kterou lze kontaktovat v případě nemožnosti přihlášení a tak dále.

Nyní, když vytvoříme nové připojení k přepínači, měla by se nám zobrazit zpráva: Vitej u sve diplomky.

- **Enable**: přepne kontext do administrativního režimu.
- **Configure**: přepne kontext do konfiguračního režimu.
- **Banner motd ***: nastaví banner „zpráva dne“.
- **Exit**: vystoupí z konfiguračního režimu.

3.2.3 Nastavení protokolu OpenFlow

Pro úspěšné nastavení OpenFlow protokolu je nutné se nacházet v konfiguračním režimu. Do konfiguračního prostředí OpenFlow se přejde pomocí klíčového slova „openflow“. Maximálně může Aruba podporovat až 128 instancí OpenFlow protokolu. Následuje ukázka konfigurace protokolu OpenFlow.

```
Aruba-3810M-48G-PoEP-1-slot# configure
Aruba-3810M-48G-PoEP-1-slot(config)# openflow
Aruba-3810M-48G-PoEP-1-slot(openflow)# openflow instance di-
pprace
Aruba-3810M-48G-PoEP-1-slot(of-inst-dipprace)# openflow con-
troller-id 1 ip 192.168.10.10 controller-interface vlan 2
Aruba-3810M-48G-PoEP-1-slot(of-inst-dipprace)# member vlan 3
Aruba-3810M-48G-PoEP-1-slot(config)# vlan 2 tagged 1
Aruba-3810M-48G-PoEP-1-slot(config)# vlan 3 tagged 3-5
```

3.3 MikroTik

Směrovač využitý k testování funkcionalit protokolu OpenFlow.

3.3.1 Technická specifikace

- Dual chain wireless 2.4GHz, Single chain wireless 5GHz
- 650MHz CPU, 64 MB of RAM
- Five x 10/100Mbps Ethernet ports
- Passive PoE output on port 5
- USB port for 3G/4G modem

(Products hAP lite, [b. r.])

3.4 Testovací nástroje

K získání a ověření výsledků jsou třeba specializované nástroje. V této práci jsou využity hlavně nástroje ICMP, iPerf a nástroje poskytující směrovačem MikroTik jako je sledování komunikace na rozhraních, či protokol TraficFlow.

3.4.1 ICMP

Internet Control Message Protocol je základní protokol síťové vrstvy TCP/IP. ICMP je považován za součást protokolu IP. To znamená, že ICMP je implementován na jakémkoliv zařízení, které využívá TCP/IP. ICMP je využíván hlavně jako služební protokol k zasílání zpráv, díky kterým je možné informovat jednotlivé sítě o různých situacích.

ICMP podporuje tyto typy zpráv:

- 0: Echo odpověď (využívá se s typem 8, Ping žádost),
- 3: Cíl je nedostupný,
- 4: Požadavek o snížení rychlosti odesílajícího zařízení,
- 5: Přeměřování,
- 8: Echo požadavek (využívá se s typem 0, Ping odpověď),
- 9: Router Advertisement,
- 10: Router Solicitation,
- 11: Překročení času,
- 12: Problém s parametry,
- 13 Požadavek o časové razítko (využívá se s typem 14),
- 14: Odpověď s časovým razítkem (využívá se s typem 13),
- 17: Požadavek o masku adresy (využívá se s typem 18),
- 18: Odpověď s maskou adresy (využívá se s typem 17).

Vzhledem k zaměření této práce, je využit hlavně typ zpráv 0 a 8. Ping se využívá zejména k ověření konektivity a k ověření kvality připojení. Pracuje na principu, kdy jedno zařízení vygeneruje zprávu Echo request a zašle ji jinému zařízení. To, pokud je zpráva určená jemu, vygeneruje odpověď a zašle ji zpět původnímu odesílateli. Kromě ověření konektivity, je ověřena i přibližná kvalita připojení. Dále se eviduje i počet skoků (TTL) a čas kdy zpráva byla odeslána a čas kdy byla přijata odpověď. Lze tedy určit i čas, jak dlouho trvalo zprávě se zprávou „dojít“ k cíli a zpátky. Tento časový interval se nazývá obousměrná latence. V případě cesty, kde je více přepínačů, není tato metoda stoprocentní. Neznáme z této zprávy přepínač, kde se zpráva opozdil. Nicméně převládá jednoduchost využití tohoto nástroje. (Network Layer/Internet Protocols, 2002)

3.4.2 iPerf3

iPerf3 je nástroj, který slouží k měření maximální možné šířky pásma na IP síti. Podporuje širokou škálu parametrů spojených s časováním, bufferováním a protokoly jako TCP, UDP, SCTP s IPv4 a IPv6. Každý test podá zprávu o šířce pásma, ztrátě a dalších parametrech. iPerf3 je vyvinut společností ESnet a Lawrence Berkeley National Laboratory (původně byl iPerf3 vyvíjen laboratoří NLANR/DAST). Jedná se o cross-platform nástroj. Lze sním pracovat na různých platformách včetně: Windows, Linux, Android, MacOS, FreeBSD, OpenBSD a dalších.

iPerf nabízí následující funkcionality:

- TCP a SCTP:
    - měření šířky pásma,
    - zpráva o MSS/MTU velikosti,
    - podpora pro velikost TCP okna přes socket buffer.
- UDP:
    - klient může vytvořit UDP stream pro specifikovanou šířku pásma,
    - měřit ztrátu paketů,
    - měřit zpoždění paketů,
    - schopnost multicastu.

(iPerf - The ultimate speed test tool for TCP, UDP and SCTP, [b. r.])

4 PRAKTICKÁ ČÁST

Tato kapitola je zaměřená na zapojení a konfiguraci topologie, zahrnující přepínač Aruba 3810M od společnosti HPE a směrovač hAP ac lite (RB952Ui -5ac2nD) od společnosti MikroTik, včetně implementace protokolu OpenFlow a měření 2 režimů komunikace mezi hosty. Hlavní cíl je porovnat rychlost komunikace mezi využitím softwarových tabulek a hardwarových tabulek na přepínači HPE Aruba. Testování je provedeno v režimu, kdy jsou tabulky prázdné (pouze s Flow pravidly, které jsou potřebné pro komunikaci hostů) a v režimu kdy jsou plné (zaplněné náhodně generovanými Flow pravidly).

Jedná se o 2 typy pamětí, do kterých si přepínač Aruba ukládá Flow pravidla. Oba typy jsou fyzické (hardwarové) prvky integrované v zařízení Aruba. Hardwarové tabulky využívají speciální typ paměti zvaný ASIC[12]. Do tohoto typu paměti se za pomoci procesoru, který zajišťuje komunikaci s kontrolérem, uloží veškeré potřebné komponenty pro chod komunikace. Následně již není procesor využíván. ASIC tedy není v zásadě pouze typ paměti, ale jedná se specializovaný čip, který umí sám vyhledávat, upravovat a porovnávat zprávy s pravidly a následně zprávu přesměrovat na port.[13]

Oproti tomu softwarové tabulky si ukládají pravidla do klasické operační paměti typu RAM, kde pro veškeré operace nad příchozí zprávou (vyhledávání, ukládání, modifikace) je třeba CPU.

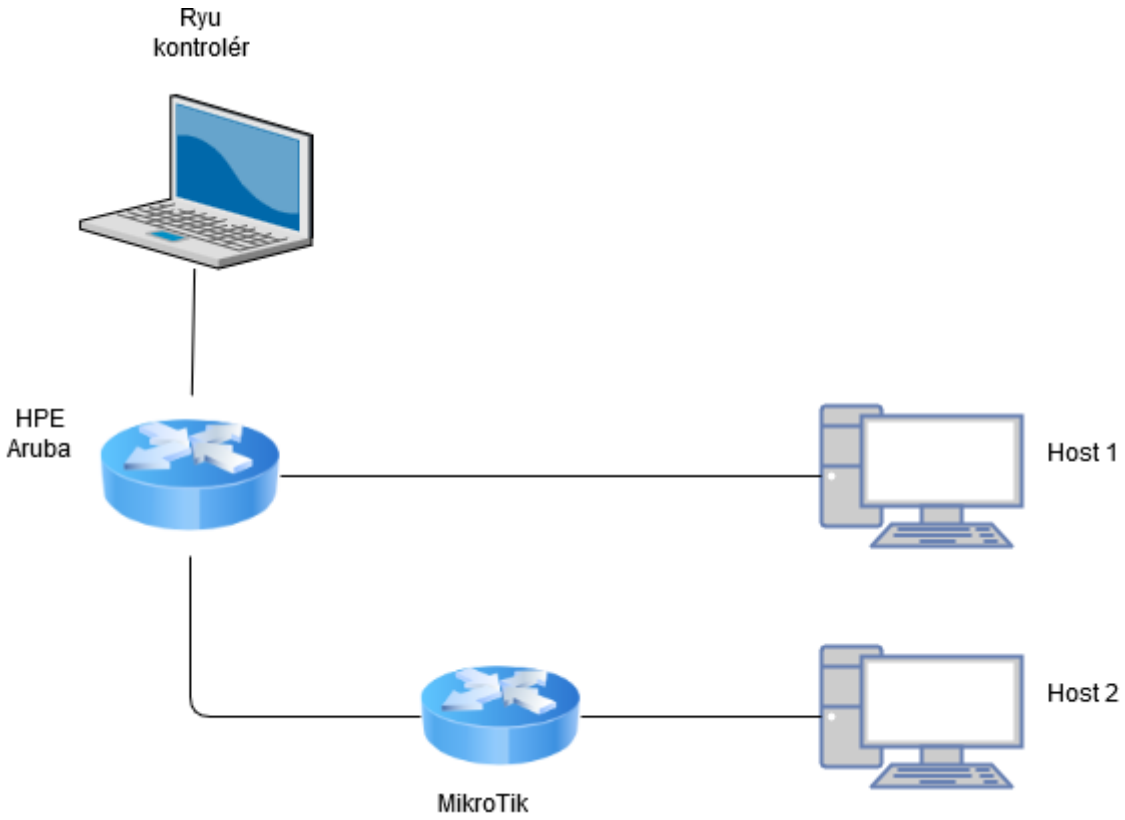
Z těchto rozdílů u jednotlivých typů pamětí, lze vyvodit předpoklad, že HW paměti budou nepopsatelně rychlejší, nicméně jejich kapacity budou znatelně menší kvůli náročnosti na výrobu a ceně, která se od toho odvíjí.

[12] ASIC – integrovaný obvod, který pro běžný chod nepotřebuje CPU (existují nicméně situace kdy jej potřebuje, například neumí zažádat kontrolér o nová pravidla, k tomu je třeba CPU).

[13] Do chvíle, dokud má odpovídající pravidla, pokud přijde zpráva, které neodpovídá žádné pravidlo, je třeba opět využít CPU.

4.1 Topologie

Celá infrastruktura se skládá z přepínače HPE Aruba, směrovače MikroTik, 2 hostů se systémem Windows 10 a kontroléru Ryu běžícím na systému Linux Ubuntu, který je umístěn na virtuálním stroji v prostředí VirtualBox.



Obrázek 4: Topologie zapojení

Zdroj: vlastní

Z obrázku (Obrázek 4) je patrné, že hlavní komunikační tok zahrnuje přepínač Aruba, který dostává Flow pravidla od kontroléru Ryu. MikroTik je využit pro testovací účely, které nabízí.

Tabulka 1: Adresovací tabulka

Ryu kontrolér	192.168.10.10 /24
VLAN 2	192.168.10.11 /24
VLAN 3	192.168.0.10 /24
Host 1	192.168.0.1 /24
Host 2	192.168.0.2 /24

Zdroj: vlastní

4.2 Konfigurace HPE Aruba 380M

Bez žádné konfigurace se defaultně HPE Aruba chová jako běžný L2 přepínač, který pracuje na základě porovnávání MAC adres. Jak již bylo zmíněno výše, CLI se velmi podobá příkazové řádce, kterou využívá i společnost Cisco.

K propojení mezi počítačem, na kterém bude probíhat konfigurace Aruby je využit sériový port COM4 na portu 2244 a nástroj Putty.

4.2.1 Konfigurace VLAN

Aruba má defaultně veškeré porty ve VLAN 1. V rámci protokolu OpenFlow je třeba nakonfigurovat porty do různých VLAN v závislosti na tom, který port bude připojený na kontrolér, který k hostům a které mají zůstat v klasickém módu pro L2 přepínač. Mimo jiné je třeba nakonfigurovat IP adresy pro tyto VLANy. (Aruba jinak nemůže navázat spojení s kontrolérem, a to i když se v konfiguraci protokolu OpenFlow nastavuje ID a IP adresa kontroléru).

Pro nastavení VLAN je třeba přejít do konfiguračního módu. Defaultně je tento mód nastaven bez hesla. To se provádí příkazem:

```
Aruba-3810M-48G-PoEP-1-slot# configure
```

Nejdříve je třeba odebrat porty, které jsou přidělené do VLAN 1. Přepnutí do konfiguračního modu VLAN 1 se provede následujícím příkazem.

```
Aruba-3810M-48G-PoEP-1-slot(config)# vlan 1
```

Pro lepší možnost testování, je vyhrazeno pro testování 5 portů (očíslovaných 1 - 5), místo potřebných 3.

Odebrání se provede pomocí příkazu:

```
Aruba-3810M-48G-PoEP-1-slot(config)# vlan 1 untagged 1-5
```

Dále je třeba vytvořit VLAN, která bude obsahovat port pro komunikaci s kontrolérem. Příkaz je stejný jako u vstupu do konfiguračního módu u VLAN 1. Aruba sama vytvoří VLAN, pokud nedokáže nalézt zadávané ID.

```
Aruba-3810M-48G-PoEP-1-slot(config)# vlan 2
```

Přidělit port a nastavit jej jako Tagged.

```
Aruba-3810M-48G-PoEP-1-slot(config)# vlan 2 tagged 1
```

Jako poslední následuje nastavení IP adresy, podle tabulky 1 – Adresace topologie.

```
Aruba-3810M-48G-PoEP-1-slot(config)# vlan 2 ip 192.168.10.11
```

Stejná sekvence příkazů následuje i pro nastavení VLAN pro komunikaci hostů, kterou bude protokol OpenFlow využívat.

```
Aruba-3810M-48G-PoEP-1-slot(config)# vlan 3
Aruba-3810M-48G-PoEP-1-slot(config)# vlan 3 tagged 2-5
Aruba-3810M-48G-PoEP-1-slot(config)# vlan 3 ip 192.168.0.10
```

4.3 Konfigurace protokolu OpenFlow

Nejdříve se použije příkaz pro vstup do konfiguračního módu OpenFlow. To se provede:

```
Aruba-3810M-48G-PoEP-1-slot# configure
Aruba-3810M-48G-PoEP-1-slot(config)# openflow
```

Následně je třeba vytvořit instanci OpenFlow, která se bude později spouštět.

```
Aruba-3810M-48G-PoEP-1-slot(openflow)# openflow instance dipprace
```

Dále je třeba nakonfigurovat ID kontroléru, IP adresu kontroléru a rozhraní (VLAN 2), na které je kontrolér připojen.

```
Aruba-3810M-48G-PoEP-1-slot(of-inst-dipprace)# openflow controller-id 1 ip 192.168.0.10 controller-interface vlan 2
```

Konfigurace portů (VLAN 3), na kterých jsou připojení hosté, se provede:

```
Aruba-3810M-48G-PoEP-1-slot(of-inst-dipprace)# member vlan 3
```

Dále je třeba provázat vytvořený kontrolér s instancí protokolu OpenFlow.

```
Aruba-3810M-48G-PoEP-1-slot(config)# openflow instance dipprace
controller-id 1
```

Nastavení verze protokolu OpenFlow.

```
Aruba-3810M-48G-PoEP-1-slot(openflow)# openflow instance di-
pprace version 1.3
```

Nastavení defaultní akce pro zahození zprávy, pokud neodpovídá žádné Flow pravidlo.

```
Aruba-3810M-48G-PoEP-1-slot(openflow)# openflow instance di-
pprace default-miss-action drop
```

A nakonec spuštění instance protokolu OpenFlow.

```
Aruba-3810M-48G-PoEP-1-slot(config)# openflow instance dipprace
enable
```

4.4 Konfigurace kontroléru Ryu

Ryu kontrolér je jednoduše skript napsaný v jazyce Python. O konfiguraci a spuštění tohoto skriptu se stará modul jménem Ryu-manager. Tomuto modulu se předá parametrizace a výše zmíněný skript a Ryu-manager se spustí jako server, či deamon, který naslouchá na určitém portu a reaguje na požadavky přepínače, podle implementace, která je popsána ve skriptu. V této práci je využit skript s názvem SimpleSwith13. Ryu skript SimpleSwitch13 má několik metod a anotací. (simple_switch_13, 2017)

```
def __init__(self, *args, **kwargs):
```

Tento konstruktor proběhne při spuštění kontroléru. Inicializuje SimpleSwitch13 a čeká na připojení kontroléru.

```
def switch_features_handler(self, ev):
```

Po připojení přepínače ke kontroléru proběhne metoda switch_features_handler. Toto je zajištěné díky anotaci:

```
@set_ev_cls(ofp_event.EventOFPSwitchFeatures, CONFIG_DISPAT-
CHER)
```

Vstupní parametr metody switch_features_handler self patří k návrhovému vzoru programovacího jazyka Python. Parametr „ev“ je datová struktura zapouzdřující informace o přepínači jako je jeho identifikátor, či parser, díky kterému je možné zasílat zprávy zpět přepínači.

Defaultně tato metoda vkládá první pravidlo, které zasílá zprávy na kontrolér, pokud přepínač nemá odpovídající pravidlo.

```
def add_flow(self, datapath, priority, match, actions, buffer_id=None):
```

Add_flow zajišťuje vkládání pravidel do přepínače pomocí metody send_msg, do které vstupuje objekt obsahující zparsovanou zprávu pomocí metody OFPFlowMod. OFPFlowMod parsuje několik parametrů.

- Datapath – ID přepínače.
- Priority - číslo určující prioritu vkládaného pravidla.
- Match - pravidlo pro porovnání.
- Table-id – ID flow tabulky, do které se má pravidlo vložit.
- Instructions – akce která se má se zprávou vykonat.

```
@set_ev_cls(ofp_event.EventOFPPacketIn, MAIN_DISPATCHER)
```

Tato anotace zajišťuje zavolání metody _packet_in_handler a to vždy, když přijde nová zpráva na kontrolér z přepínače.

```
def _packet_in_handler(self, ev):
```

Parametry self a ev mají stejnou funkcionalitu jako u metody switch_features_handler. Díky této metodě má kontrolér schopnost se učit nové MAC adresy, konkrétně se to zajistí pomocí datové struktury mac_to_port. Pomocí parser.OFPActionOutput(out_port) se vloží port vedoucí k cílovému zařízení do proměnné actions, které se následně s ostatními parametry předají metodě add_flow.

Následuje ukázka, jak aktivovat Ryu kontrolér pomocí Ryu-manager v prostředí Linux Ubuntu.

```
ubuntu@sdnhubvm:~/ryu[12:42] (master)$ cd /home/ubuntu/ryu && ./bin/ryu-manager --verbose --ofp-tcp-listen-port 6633 ryu/app/simple_switch_13.py
```

Parametr --ofp-tcp-listen-port konfiguruje port, na kterém bude Ryu kontrolér naslouchat. Pokud se kontroléru nepodaří navázat na žádném rozhraní komunikaci, pokusí se připojit na localhost (127.0.0.1), který zpravidla končí chybovou hláškou. Nicméně i po výpisu této chyby Ryu kontrolér stále naslouchá na přiřazeném rozhraní a přiděleném portu.

4.5 Testování komunikace – hardwarové tabulky

Tato kapitola je zaměřená na ověření komunikace a testování rychlosti režimu OpenFlow. Pro tyto účely jsou využívány výše popsané nástroje jako ICMP, iPerf3, WireShark a nástroje které podporuje směrovač MikroTik, jako je sledování komunikace na portech, či nástroj TraficFlow. Jak již bylo zmíněno, testování proběhne ve 2 různých módech ukládání Flow pravidel na přepínači Aruba. Jedná se o softwarové a hardwarové tabulky. Nejdříve je provedeno měření na hardwarových tabulkách. Pro nastavení konfigurace přepínače Aruba, aby ukládala Flow pravidla pouze do hardwarové tabulky, je třeba vykonat následující sekvenci příkazů.[14]

```
Aruba-3810M-48G-PoEP-1-slot# configure

Aruba-3810M-48G-PoEP-1-slot(config)# openflow instance dipprace
disable

Aruba-3810M-48G-PoEP-1-slot(config)# openflow instance dipprace
table-num policy-table 0
```

Nyní je třeba upravit skript SimpleSwitch13, aby kontrolér věděl do jaké tabulky, s jakým ID má pravidla ukládat. Úprava se týká funkce „add_flow“, kde je třeba upravit parametr table_id na 0.

```
if buffer_id:
    mod = parser.OFPFlowMod(datapath=datapath, buffer_id =
    buffer_id, priority=priority, match=match, table_id=0,in-
    structions=inst)
else:
```

[14] Defaultně má přepínač Aruba nastavené ID HW tabulky na 100. Další možnost konfigurace pro zajištění, že se pravidla budou ukládat pouze do HW tabulky je změna parametru table_id ve skriptu SimpleSwitch13.

```
mod = parser.OFPFlowMod(datapath=datapath, priority=prio
rity, match=match, table_id=0, instructions=inst)
```

Nyní je třeba povolit OpenFlow pomocí příkazu.

```
Aruba-3810M-48G-PoEP-1-slot(config)# openflow instance dipprace
enable
```

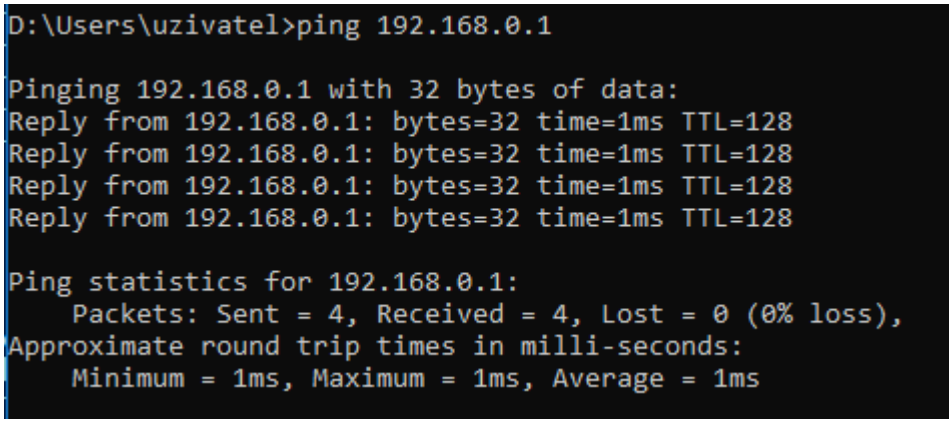
4.5.1 ICMP

Pomocí protokolu ICMP, lze velmi snadno ověřit komunikaci mezi hosty a zjistit přibližnou odezvu, za jak dlouho druhý host odpoví. Provede se příkazem.

```
ping 192.168.0.2
```

Pomocí parametru –t, lze spustit ping bez omezení počtu požadavků.

```
ping 192.168.0.2 -t
```



Obrázek 5: ICMP – PING: HW tabulky

Zdroj: vlastní

Z obrázku 5 je patrné, že spojení proběhlo úspěšně a že čas, než odpoví druhý host je zpravidla roven 1 milisekundě.

4.5.2 iPerf

iPef3 je nástroj, který dokáže změřit kromě času odezvy a ověření komunikace i propustnost dané linky a dokáže nasimulovat i více paralelních připojení najednou.

iPerf pracuje na principu klient – server. Nejdříve je třeba spustit na prvním hostovi server, který bude naslouchat na nějakém portu. To se provede příkazem.

```
./iperf3.exe -s
```

Nyní na druhém hostovi se spustí iPerf s konfigurací pro klienta

```
./iperf3.exe -c 192.168.0.2
```

V defaultním nastavení se otestuje připojení pomocí protokolu TCP.

```
PS C:\iperf3> ./iperf3.exe -c 192.168.0.2
Connecting to host 192.168.0.2, port 5201
[  4] local 192.168.0.1 port 49964 connected to 192.168.0.2 port 5201
[ ID] Interval           Transfer     Bandwidth
[  4]   0.00-1.00   sec   112 MBytes   943 Mbits/sec
[  4]   1.00-2.00   sec   112 MBytes   943 Mbits/sec
[  4]   2.00-3.00   sec   112 MBytes   942 Mbits/sec
[  4]   3.00-4.00   sec   112 MBytes   942 Mbits/sec
[  4]   4.00-5.00   sec   112 MBytes   942 Mbits/sec
[  4]   5.00-6.00   sec   112 MBytes   942 Mbits/sec
[  4]   6.00-7.00   sec   112 MBytes   942 Mbits/sec
[  4]   7.00-8.00   sec   112 MBytes   942 Mbits/sec
[  4]   8.00-9.00   sec   112 MBytes   942 Mbits/sec
[  4]   9.00-10.00  sec   112 MBytes   942 Mbits/sec
- - - - - - - - - - - - - - - - - - - - - - - - -
[ ID] Interval           Transfer     Bandwidth
[  4]   0.00-10.00  sec  1.10 GBytes   942 Mbits/sec                  sender
[  4]   0.00-10.00  sec  1.10 GBytes   942 Mbits/sec                  receiver
```

Obrázek 6:iPerf – TCP: HW tabulky

Zdroj: vlastní

Výsledek měření (obrázek 6) ukazuje, že je využit plný potenciál přepínače Aruba. Propustnost na lince se blíží 1 gigabitu, která je uvedena ve specifikaci přepínače Aruba, i směrovače MikroTik v oficiální specifikaci těchto zařízení. Stejné hodnoty byly zjištěny i v případě, kdy byl směrovač MikroTik odpojen z topologie.

Nyní se spustí stejný test, ale pomocí protokolu UDP. To se provede přidáním parametru „-u“.

```
./iperf3.exe -c 192.168.0.2 -u
```

```
PS C:\iperf3> ./iperf3.exe -c 192.168.0.2 -u
Connecting to host 192.168.0.2, port 5201
[  4] local 192.168.0.1 port 51920 connected to 192.168.0.2 port 5201
[ ID] Interval           Transfer     Bandwidth       Total Datagrams
[  4]   0.00-1.00   sec   128 KBytes  1.05 Mbits/sec  16
[  4]   1.00-2.00   sec   128 KBytes  1.05 Mbits/sec  16
[  4]   2.00-3.00   sec   128 KBytes  1.05 Mbits/sec  16
[  4]   3.00-4.00   sec   128 KBytes  1.05 Mbits/sec  16
[  4]   4.00-5.00   sec   128 KBytes  1.05 Mbits/sec  16
[  4]   5.00-6.00   sec   128 KBytes  1.05 Mbits/sec  16
[  4]   6.00-7.00   sec   128 KBytes  1.05 Mbits/sec  16
[  4]   7.00-8.00   sec   128 KBytes  1.05 Mbits/sec  16
[  4]   8.00-9.00   sec   128 KBytes  1.05 Mbits/sec  16
[  4]   9.00-10.00  sec   128 KBytes  1.05 Mbits/sec  16
- - - - - - - - - - - - - - - - - - - - - - - - -
[ ID] Interval           Transfer     Bandwidth       Jitter    Lost/Total Datagrams
[  4]   0.00-10.00  sec  1.25 MBytes  1.05 Mbits/sec  0.221 ms  0/159 (0%)
[  4] Sent 159 datagrams
```

Obrázek 7:iPerf – UDP: HW tabulky

Zdroj: vlastní

Výsledky (Obrázek 7) jsou zde odlišné z důvodu, že v základní konfiguraci iPerf vynucuje rychlost 1 Mbps. V tomto měření však bylo cílem zjistit, zda všechny zprávy dojdou k cíli. Zde je vidět, že je nulová ztráta datagramů (Lost/Total Datagrams). Pro vynucení maximální rychlosti lze iPerf3 parametrizovat pomocí přepínače -b „požadovaná rychlost“.

4.5.3 MikroTik

Zařízení MikroTik nabízí diagnostické nástroje jako je sledování komunikace na portech, či nástroj jménem TraficFlow (je obdoba protokolu NetFlow od společnosti Cisco). Vzhledem k tomu, že TraficFlow pracuje na principu odchytávání komunikace 2 různých sítích, nelze sledovat komunikaci mezi hosty, kteří jsou zapojeny v jedné sítí v režimu OpenFlow. Lze sledovat pouze komunikaci mezi kontrolérem a přepínačem Aruba, který je v jiné síti.

MikroTik nabízí GUI rozhraní pro konfiguraci. Vzhledem k tomu, že MikroTik je využit pouze jako L2 zařízení, není třeba žádná speciální konfigurace a lze rovnou pozorovat komunikaci na portech.

Pomocí dokumentace k zařízení MikroTik, se provede přihlášení do konfigurace[15].

V záložce IP v podkategorii „Interfaces“ se zobrazí aktuální vytížení portů, které se mění v reálném čase. Pro test je odeslán z jednoho hosta na druhého soubor o velikosti 3 GB.

		Name	Type	L2 MTU	Tx	Rx	Tx Packet (p/s)	Rx Packet (p/s)
- D	R	bridge2	Bridge	1598	78.3 kbps	6.1 kbps	7	5
D		ether1-gateway	Ethernet	1598	0 bps	0 bps	0	0
D	RS	ether2-master-local	Ethernet	1598	79.3 kbps	6.3 kbps	8	4
D	RS	ether3-slave-local	Ethernet	1598	0 bps	0 bps	0	0
D	RS	ether4-slave-local	Ethernet	1598	986.7 Mbps	37.3 Mbps	81 292	72 623
D	RS	ether5-slave-local	Ethernet	1598	37.3 Mbps	986.7 Mbps	72 626	81 292

Obrázek 8: MikroTik HW tabulky

Zdroj: vlastní

[15] https://i.mt.lv/cdn/rb_files/hAP-lite-qg.pdf

Zdroj: vlastní

Obrázek pochází z okamžiku, kdy probíhá přenos souboru mezi hosty. Je patrné, že přepínač Aruba a směrovač MikroTik využívají své maximální kapacity pro přenos (obě zařízení dle specifikace má dosahovat přenosu až 1Gbps).

4.6 Testování komunikace – softwarové tabulky

Pro testování softwarových tabulek, je třeba upravit konfiguraci přepínače Aruba. Změna se provede následujícími příkazy.

Vstup do konfiguračního módu.

```
Aruba-3810M-48G-PoEP-1-slot# configure
```

Deaktivování instance OpenFlow.

```
Aruba-3810M-48G-PoEP-1-slot(config)# openflow instance dipprace
disable
```

Změna ID HW tabulky na 1.

```
Aruba-3810M-48G-PoEP-1-slot(config)# openflow instance dipprace
table-num policy-table 1
```

Změna ID SW tabulky na 0.

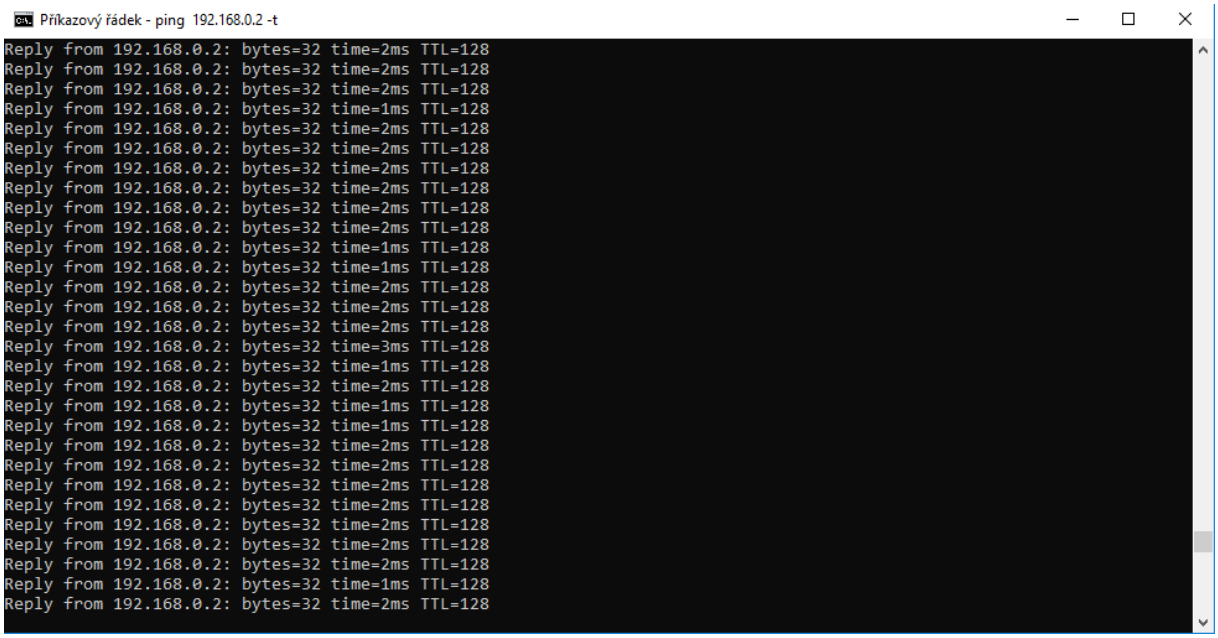
```
Aruba-3810M-48G-PoEP-1-slot(config)# openflow instance dipprace
table-num sw-table 0
```

Nyní bude přepínač Aruba ukládat veškeré flow pravidla do softwarové tabulky.

4.6.1 ICMP

Test pomocí protokolu ICMP bude probíhat na stejném principu, jako u hardwarových tabulek.

```
ping 192.168.0.2
```



Obrázek 9: ICMP – PING: SW tabulky

Zdroj: vlastní

Z obrázku 9 je zřejmé, že hosté dokáží spolu komunikovat.

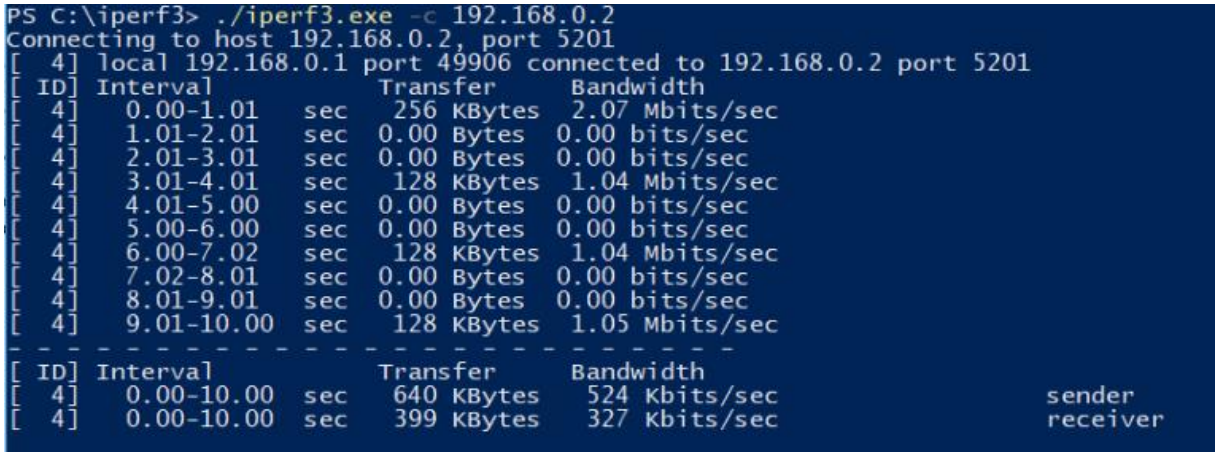
4.6.2 iPerf

Nastavení nástroje iPerf je totožné s nastavením u hardwarových tabulek. Strana serveru.

```
./iperf3.exe -s
```

Strana klienta pro protokol TCP.

```
./iperf3.exe -c 192.168.0.2
```



Obrázek 10: iPerf – TCP: SW tabulky

Zdroj: vlastní

Zde je vidět značný propad rychlosti komunikace (Obrázek 10). Z řádu stovek MB/s, je propad do řádu stovek kilobitů. Každé 2 sekundy je dokonce 0 propustnost. Je patrné, že využití softwarových tabulek, není optimální pro využití v běžné síti.

Pro test s využitím protokolu UDP se využije opět příkaz.

```
./iperf3.exe -c 192.168.0.2 -u
```

```
iperf Done.
PS C:\iperf3> ./iperf3.exe -c 192.168.0.2 -u
Connecting to host 192.168.0.2, port 5201
[  4] local 192.168.0.1 port 61319 connected to 192.168.0.2 port 5201
[ ID] Interval           Transfer     Bandwidth       Total Datagrams
[  4]   0.00-1.01   sec   128 KBytes  1.04 Mbits/sec  16
[  4]   1.01-2.01   sec   136 KBytes  1.11 Mbits/sec  17
[  4]   2.01-3.01   sec   120 KBytes   983 Kbits/sec  15
[  4]   3.01-4.02   sec   136 KBytes  1.11 Mbits/sec  17
[  4]   4.02-5.00   sec   120 KBytes   998 Kbits/sec  15
[  4]   5.00-6.00   sec   128 KBytes  1.05 Mbits/sec  16
[  4]   6.00-7.00   sec   128 KBytes  1.05 Mbits/sec  16
[  4]   7.00-8.01   sec   128 KBytes  1.05 Mbits/sec  16
[  4]   8.01-9.00   sec   128 KBytes  1.05 Mbits/sec  16
[  4]   9.00-10.01  sec   136 KBytes  1.11 Mbits/sec  17
- - - - - - - - - - - - - - - - - - - - - - - - -
[ ID] Interval           Transfer     Bandwidth       Jitter    Lost/Total Datagrams
[  4]   0.00-10.01  sec  1.26 MBytes  1.05 Mbits/sec  0.872 ms  30/159 (19%)
[  4] Sent 159 datagrams
```

Obrázek 11: iPerf – UDP: SW tabulky

Zdroj: vlastní

Dostaneme podobné výsledky, jako u testu s využitím protokolu TCP. Potvrzuje to výsledek, že tento mód zapojení se nehodí pro běžnou komunikaci. Je zjištěna 19% ztráta datagramů.

U testu s využitím protokolu UDP je možnost nastavit vynucenou rychlost odesílání datagramů z klienta. Za parameter „-b“ lze napsat rychlost, které se má host snažit docílit. To lze provést pomocí příkazu.

```
./iperf3.exe -c 192.168.0.2 -u -b 1000 MB
```

Nicméně na straně serveru přichází stejné množství dat, jako v prvním testu s využitím UDP protokolu. To znamená, že ani vynucená rychlost připojení, není dostačující, aby přepínač Aruba přeposílal zprávy.

4.6.3 MikroTik

Přihlášení a zobrazení portů, se provede stejně jako u hardwarových tabulek. A opět se provede přenos souboru mezi hosty.

		Name	Type	L2 MTU	Tx	Rx	Tx Packet (p/s)	Rx Packet (p/s)
- D	R	bridge2	Bridge	1598	77.7 kbps	5.7 kbps	8	4
D		ether1-gateway	Ethernet	1598	0 bps	0 bps	0	0
D	RS	ether2-master-local	Ethernet	1598	78.6 kbps	6.3 kbps	9	4
D	RS	ether3-slave-local	Ethernet	1598	512 bps	0 bps	1	0
D	RS	ether4-slave-local	Ethernet	1598	354.4 kbps	6.0 kbps	33	9
D	RS	ether5-slave-local	Ethernet	1598	7.0 kbps	353.4 kbps	11	31

Obrázek 12: MikroTik: SW tabulky

Zdroj: vlastní

I zde je vidět, že maximální rychlost komunikace, je mnohonásobně snížena. Opět je potvrzena hypotéza, že softwarové tabulky, se nehodí pro běžný režim komunikace.

4.7 Shrnutí testů SW a HW tabulek

Z testů na softwarových tabulkách je zřejmé, že se nehodí pro běžný druh komunikace. Lze ji využít například pro konfiguraci síťových prvků, či pro troubleshooting, debug, kontrolu sítě, či filtraci provozu. Nicméně pro běžný druh komunikace je silně nedostačující, na rozdíl od hardwarových tabulek, které využívají maximálních kapacit přepínače Aruba.

4.8 Konfigurace Ryu – plné Flow tabulky

Následující testy se týkají opět softwarových a hardwarových tabulek, ale za předpokladu, že tabulky jsou naplněny pravidly. Pro testování s naplněnými Flow tabulkami, je potřebné upravit skript SimpleSwitch13. Jelikož v rámci této práce, není k dispozici dostatečné množství reálných hostů, byla vygenerována náhodná pravidla, která v reálném provozu nebudou fungovat, nicméně zaplní kapacitu Flow tabulky. Úprava se týká metody switch_features_handler.

```
i = 1

        while i < 500:

            self._generate_flows(ev)

            i += 1
```

Konstanta 500 určuje, kolik pravidel se vloží Flow tabulky. Hardwarové tabulky podporují přibližně 512 pravidel.[16] To znamená, že Flow tabulky nebyly zcela zaplněny. Nicméně je tento

[16] Tato konstanta není uvedená v oficiální dokumentaci, ale je výsledkem testování, kdy byl vkládán různý počet pravidel a poté bylo sledováno, zda Flow tabulka přetekla, či nikoli.

rozdíl zanedbatelný a nadále budou uváděny jako „zaplněné tabulky“. Toto opatření bylo zavedeno z důvodu, aby na přepínači Aruba zbyl prostor pro pravidla, která jsou důležitá pro testování komunikace (pravidla pro komunikaci hostů, defaultní pravidlo).

Metoda _generate_flows je připravena ke generování Flow pravidel 2 typů, První typ obsahuje zdrojovou MAC adresu, cílovou MAC adresu, zdrojovou IPv6 adresu a cílovou IPv6 adresu. Druhý typ obsahuje pouze zdrojový TCP port a cílový TCP port.

Během testování vkládání pravidel do tabulek bylo kromě počtu pravidel (512) zjištěno, že nezáleží na velikosti vkládaného pravidla. Ať byl vkládán první nebo druhý typ pravidla, vždy se jich vešlo 512. Přepínač Aruba má nejpravděpodobněji pevně danou velikost jednoho záznamu pro Flow tabulku.

Kromě této úpravy je dále potřebné implementovat funkce pro generování náhodných MAC adres, IPv6 adres a TCP portů.

```
def _random_IPv6(self):
```

Metoda generuje IPv6 adresu.

```
def _random_MAC(self):
```

Metoda generuje MAC adresu.

```
def _random_TCP(self):
```

Metoda generuje TCP port.

Toto je vše pro konfiguraci kontroléru Ryu. Nyní lze spustit Ryu kontrolér, pomocí již použitého příkazu.

```
ubuntu@sdnhubvm:~/ryu[12:42] (master)$ cd /home/ubuntu/ryu && ./bin/ryu-manager --verbose --ofp-tcp-listen-port 6633 ryu/app/simple_switch_13.py
```

Po spuštění Ryu managera, začne kontrolér zasílat generovaná pravidla dle upraveného skriptu SimpleSwitch13 do přepínače Aruba.

Ukládání Flow pravidel trvá nějaký čas, který se mění za různých okolností. Mezi hlavní kritéria patří počet Flow pravidel a typ Flow tabulky (softwarová, či hardwarová) do které se mají ukládat. Do těchto okolností lze zařadit mnohem více příkladů, jako je hardwarový výkon, či propustnost portu na kontroléru.

V tomto případě se ukládalo přibližně 500 pravidel 65 sekund.

4.9 Konfigurace přepínače Aruba – plné HW tabulky

Konfigurace se týká pouze změny id Flow tabulky, do které má přepínač Aruba ukládat pravidla, která přijdou od kontroléru.

Nejdříve je třeba vypnout aktivní instanci OpenFlow pomocí příkazu.

```
Aruba-3810M-48G-PoEP-1-slot# configure
```

```
Aruba-3810M-48G-PoEP-1-slot(config)# openflow instance dipprace disable
```

Nyní se přepne do konfiguračního módu OpenFlow

```
Aruba-3810M-48G-PoEP-1-slot(config)# openflow
```

A pomocí následující sekvence příkazů se nastaví ID hardwarové tabulky na 0 a ID softwarové tabulky na 2.

```
Aruba-3810M-48G-PoEP-1-slot(config)# openflow instance dipprace table-num sw-table 2
```

```
Aruba-3810M-48G-PoEP-1-slot(config)# openflow instance dipprace table-num policy-table 0
```

Z příkazů je patrné, že je třeba nejdříve překonfigurovat id softwarové tabulky. To je z důvodů, že přepínač Aruba požaduje rostoucí ID Flow tabulek.

4.10 Testování komunikace – naplněné HW tabulky

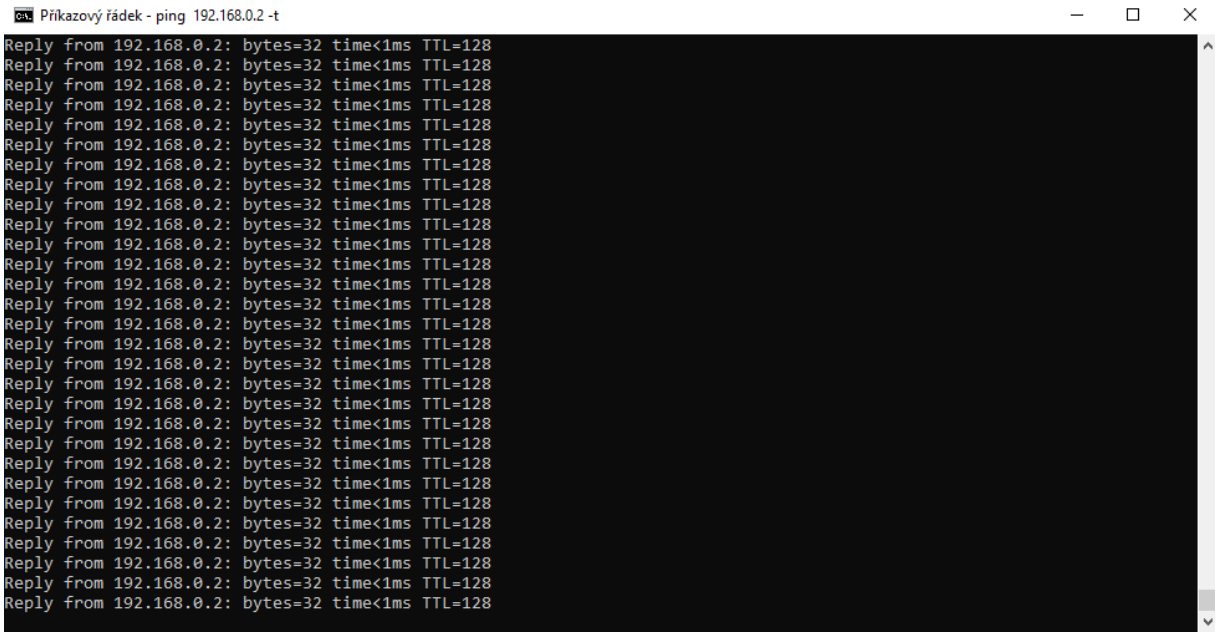
Tato podkapitola je zaměřena na testování naplněných HW tabulek. Cílem je zjistit, zda naplněné tabulky neomezují komunikaci z důvodu vyhledávání Flow pravidla, pro příchozí zprávu. Vzhledem ke specializovanému typu paměti ASIC, je přepokládáno že nebude zaznamenán výraznější pokles v propustnosti komunikace. Testování proběhne pomocí stejných nástrojů, které byly využity i u testování softwarových a hardwarových tabulek, když nebyly zcela naplněny. Jsou to nástroje iPerf3, protokol ICMP a směrovač MikroTik. Bude testována konektivita, propustnost a bude provedeno porovnání, zda naplněné Flow tabulky mají nějaký vliv na komunikaci oproti komunikaci, kde jsou Flow tabulky téměř prázdné.

4.10.1 ICMP

Pro ověření komunikace pomocí protokolu ICMP je využit příkaz.

```
ping 192.168.0.2
```

Z následujícího obrázku (Obrázek 12) je zřejmě, že hosté dokáží spolu komunikovat, a že čekání na odpověď je menší než 1 milisekunda.



Obrázek 13: ICMP – PING: plné HW tabulky

Zdroj: vlastní

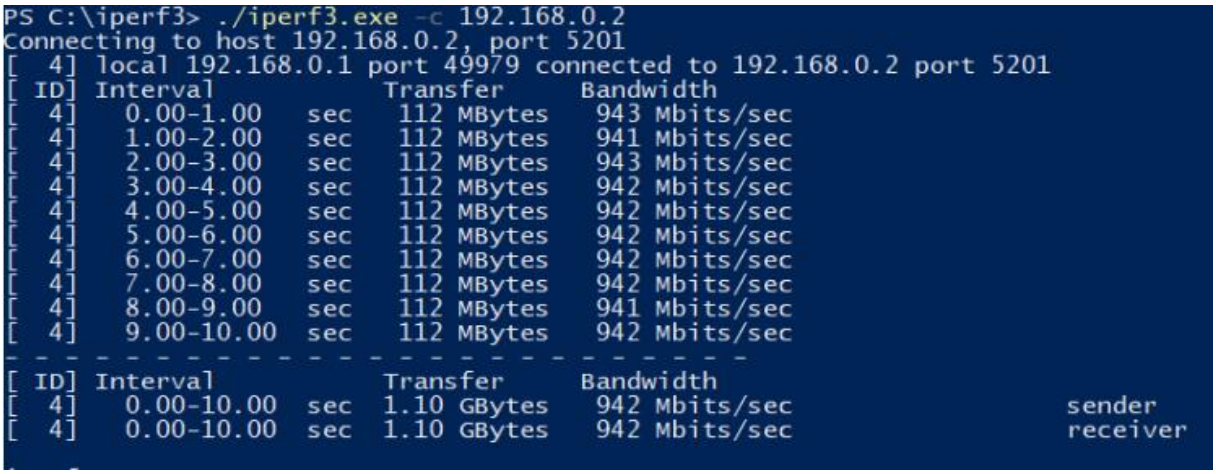
4.10.2 iPerf

Nakonfiguruje se strana serveru, která bude naslouchat.

```
./iperf3.exe -s
```

A strana klienta, která bude vysílat pakety pomocí TCP protokolu.

```
./iperf3.exe -c 192.168.0.2
```

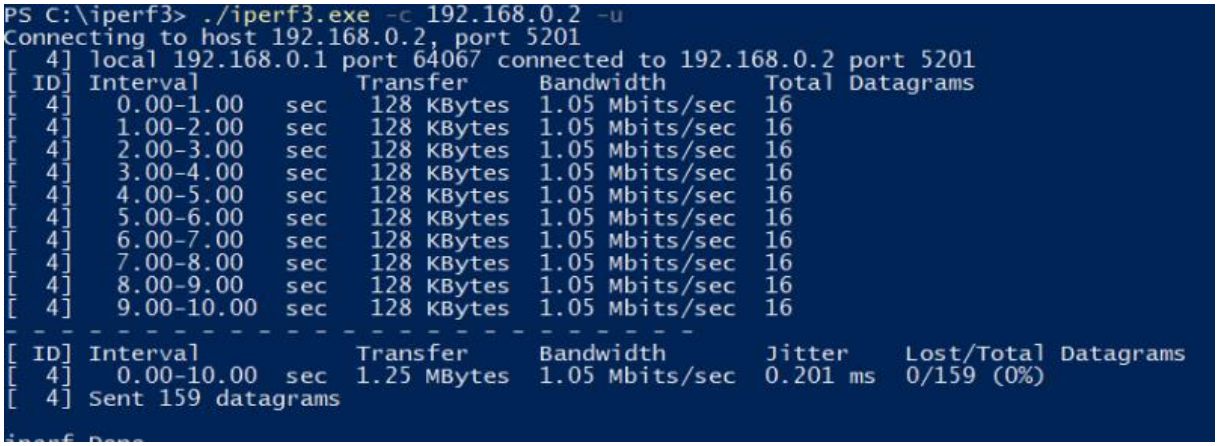


Obrázek 14: iPerf – TCP: plné HW tabulky

Zdroj: vlastní

Zde je vidět, že ačkoliv jsou HW tabulky zaplněné, tak přepínač Aruba umí efektivně v tabulkách vyhledávat a využívat správně pravidla. Přepínač Aruba pracuje s maximální svojí kapacitou.

Obrázek 15 ukazuje výsledky s využitím UDP protokolu.



Obrázek 15: iPerf – UDP: plné HW tabulky

Zdroj: vlastní

4.10.3 MikroTik

Při přenosu souboru z jednoho hosta na druhého za pomocí směrovače MikroTik je vidět, že je využit plný potenciál přepínače Aruba.

		Name	Type	L2 MTU	Tx	Rx	Tx Packet (p/s)	Rx Packet (p/s)
D	R	bridge2	Bridge	1598	78.3 kbps	6.1 kbps	7	5
D		ether1-gateway	Ethernet	1598	0 bps	0 bps	0	0
D	RS	ether2-master-local	Ethernet	1598	79.3 kbps	6.3 kbps	8	4
D	RS	ether3-slave-local	Ethernet	1598	0 bps	0 bps	0	0
D	RS	ether4-slave-local	Ethernet	1598	986.7 Mbps	37.3 Mbps	81 292	72 623
D	RS	ether5-slave-local	Ethernet	1598	37.3 Mbps	986.7 Mbps	72 626	81 292

Obrázek 16: MikroTik: plné HW tabulky

Zdroj: vlastní

4.11 Konfigurace pro SW tabulky

Pro testování naplněných softwarových Flow tabulek, je třeba upravit konfiguraci na přepínači Aruba.

Deaktivuje se instance OpenFlow.

```
Aruba-3810M-48G-PoEP-1-slot(config)# openflow instance dipprace disable
```

A provede se následující sekvence příkazů, která změní ID tabulek tak, aby přepínač Aruba ve své pipeline ukládal Flow pravidla pouze do softwarové tabulky.

```
Aruba-3810M-48G-PoEP-1-slot(config)# openflow instance dipprace table-num policy-table 1
```

```
Aruba-3810M-48G-PoEP-1-slot(config)# openflow instance dipprace table-num sw-table 0
```

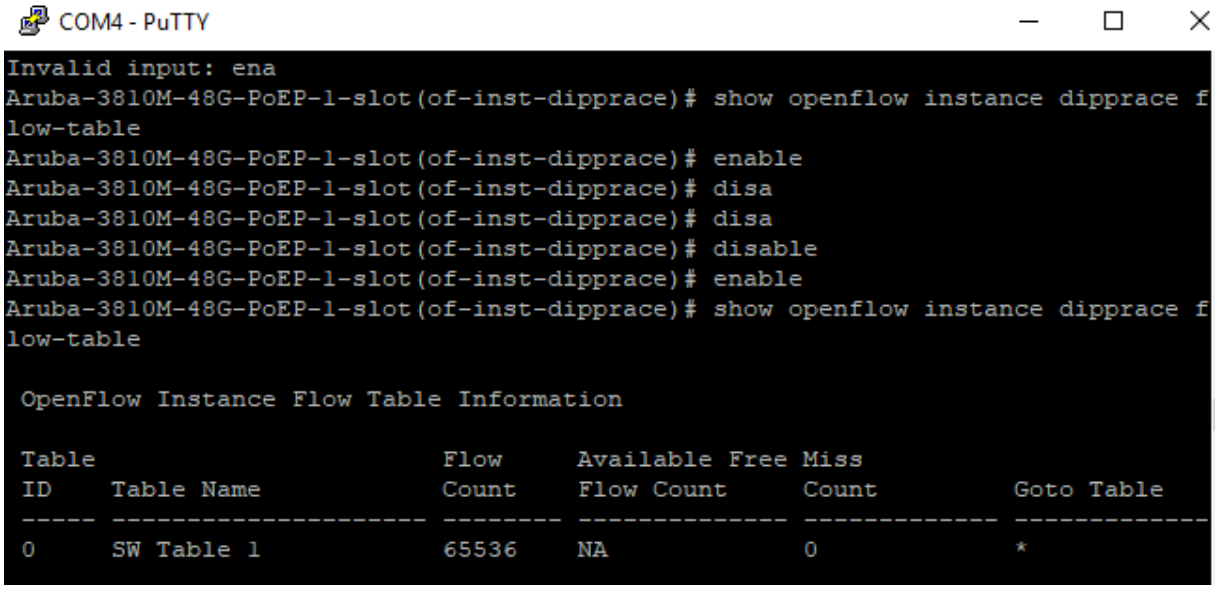
Dále je třeba upravit skript SimpleSwitch13. Úprava se týká konstanty, která určuje kolik pravidel se má uložit do Flow tabulky. Z testování vyšlo, že jich nemá být více než 65 536, jinak přepínač Aruba zasílá na Ryu kontrolér chybové hlášky, že Flow tabulka, je již plná a další příchozí pravidla budou zahozeny.

```
i = 1

        while i < 65000:

            self._generate_flows(ev)

            i += 1
```

Po restartu kontroléru lze na přepínači Aruba vypsat Flow tabulku pro ověření, zda Ryu kontrolér korektně naplnil Flow tabulku.

```
Aruba-3810M-48G-PoEP-1-slot(config)# show openflow instance di-
pprace flow-table
```

I zde byl měřen čas, jak dlouho trvalo uložení Flow pravidel. Zde bylo ukládáno přes 65 000 Flow pravidel, jejichž zápis trval kolem 132 sekund. V porovnání se zápisem do hardwarové tabulky, je toto řádově rychlejší. U hardwarových tabulek, trvalo uložení 500 pravidel 65 sekund.



Obrázek 17: Ukázka SW Flow tabulky

Zdroj: vlastní

4.12 Testování komunikace – naplněné SW tabulky

Testování SW tabulek proběhlo na stejném principu jako u hardwarových tabulek.

4.12.1 ICMP

Test pomocí protokolu ICMP potvrdil konektivitu a vypsal čas, za jak dlouho reagoval druhý klient.

```
Příkazový řádek - ping 192.168.0.2 -t
Reply from 192.168.0.2: bytes=32 time=2ms TTL=128
Reply from 192.168.0.2: bytes=32 time=2ms TTL=128
Reply from 192.168.0.2: bytes=32 time=2ms TTL=128
Reply from 192.168.0.2: bytes=32 time=2ms TTL=128
Reply from 192.168.0.2: bytes=32 time=2ms TTL=128
Reply from 192.168.0.2: bytes=32 time=2ms TTL=128
Reply from 192.168.0.2: bytes=32 time=2ms TTL=128
Reply from 192.168.0.2: bytes=32 time=2ms TTL=128
Reply from 192.168.0.2: bytes=32 time=2ms TTL=128
Reply from 192.168.0.2: bytes=32 time=2ms TTL=128
Reply from 192.168.0.2: bytes=32 time=2ms TTL=128
Reply from 192.168.0.2: bytes=32 time=2ms TTL=128
Reply from 192.168.0.2: bytes=32 time=2ms TTL=128
Reply from 192.168.0.2: bytes=32 time=2ms TTL=128
Reply from 192.168.0.2: bytes=32 time=2ms TTL=128
Reply from 192.168.0.2: bytes=32 time=2ms TTL=128
Reply from 192.168.0.2: bytes=32 time=2ms TTL=128
Reply from 192.168.0.2: bytes=32 time=2ms TTL=128
Reply from 192.168.0.2: bytes=32 time=2ms TTL=128
Reply from 192.168.0.2: bytes=32 time=2ms TTL=128
Reply from 192.168.0.2: bytes=32 time=2ms TTL=128
Request timed out.
Reply from 192.168.0.2: bytes=32 time=1ms TTL=128
Reply from 192.168.0.2: bytes=32 time=2ms TTL=128
Reply from 192.168.0.2: bytes=32 time=2ms TTL=128
Reply from 192.168.0.2: bytes=32 time=1ms TTL=128
Reply from 192.168.0.2: bytes=32 time=2ms TTL=128
Reply from 192.168.0.2: bytes=32 time=2ms TTL=128
Reply from 192.168.0.2: bytes=32 time=103ms TTL=128
```

Obrázek 18: ICMP – PING: plné SW tabulky

Zdroj: vlastní

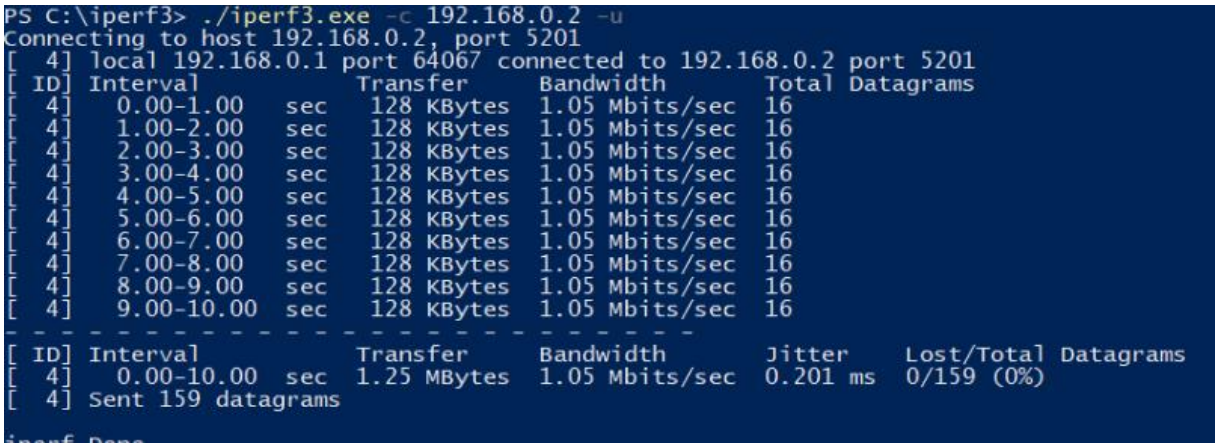
4.12.2 iPerf

Nástroj iPerf byl využit stejně jako v předchozím testování.

```
PS C:\iperf3> ./iperf3.exe -c 192.168.0.2
Connecting to host 192.168.0.2, port 5201
[  4] local 192.168.0.1 port 49919 connected to 192.168.0.2 port 5201
[ ID] Interval           Transfer     Bandwidth
[  4]   0.00-1.01   sec   256 KBytes  2.08 Mbits/sec
[  4]   1.01-2.01   sec  0.00 Bytes  0.00 bits/sec
[  4]   2.01-3.01   sec  0.00 Bytes  0.00 bits/sec
[  4]   3.01-4.01   sec   128 KBytes  1.05 Mbits/sec
[  4]   4.01-5.00   sec  0.00 Bytes  0.00 bits/sec
[  4]   5.00-6.02   sec  0.00 Bytes  0.00 bits/sec
[  4]   6.02-7.01   sec   128 KBytes  1.06 Mbits/sec
[  4]   7.01-8.01   sec  0.00 Bytes  0.00 bits/sec
[  4]   8.01-9.00   sec  0.00 Bytes  0.00 bits/sec
[  4]   9.00-10.00  sec   128 KBytes  1.05 Mbits/sec
- - - - - - - - - - - - - - - - - - - - - - - - -
[ ID] Interval           Transfer     Bandwidth
[  4]   0.00-10.00  sec   640 KBytes   524 Kbits/sec                  sender
[  4]   0.00-10.00  sec   398 KBytes   326 Kbits/sec                  receiver
```

Obrázek 19: iPerf – TCP: plné SW tabulky

Zdroj: vlastní



Obrázek 20: iPerf – UDP: plné SW tabulky

Zdroj: vlastní

Výsledky jsou obdobné, jako u měření s prázdnými softwarovými tabulkami.

4.12.3 MikroTik

Výše popsané výsledky potvrdil i směrovač MikroTik, který při odesílání souborů mezi hosty, na sledovacím nástroji, ukázal maximální vytíženost.

4.13 Shrnutí testů naplněných SW a HW tabulek

V této etapě praktické části, se potvrdilo, že hardwarové tabulky, se hodí pro běžnou komunikaci. Nicméně oproti softwarovým tabulkám, lze do hardwarových tabulkách uložit pouze 500 pravidel a uložení do těchto tabulek trvá výrazně delší čas. Je to z důvodu že HPE Aruba 3810M je převážně určen pro komunikaci, kde pracuje jako klasický L3 přepínač. OpenFlow protokol je pro toto zařízení určen spíše do menšího provozu. Nelze jej nasadit do velkých firem, či dokonce datacenter.

Výrobce nikde neuvádí, z jakých důvodů je HW tabulka o tolik menší, nicméně hlavní důvod je že HW tabulky se ukládají do specifických HW pamětí (ASIC), které mají v případě přepínače Aruba mnohem menší kapacitu, než operační paměť RAM. Další z hlavních důvodů jsou nejpravděpodobněji vysoké náklady na výrobu tohoto typu paměti.

4.14 Shrnutí

Následující tabulka porovnává veškeré naměřené hodnoty získané během měření. Porovnávají se zde nejen konektivita a propustnost, ale také doba, jak dlouho se Flow pravidla ukládala do tabulek, nebo zda mělo vliv na výkon případy, kdy byly tabulky zcela naplněné.

Tabulka 2: Výsledky měření

	HW tabulka	SW tabulka	Zaplněná HW tabulka	Zaplněná SW tabulka
ICMP	1 ms	1-2 ms	1 ms	1-2 ms
TCP	942 Mbps	1-2 Mbps	942 Mbps	1-2 Mbps
UDP	0 %	19 %	0 %	18 %
Doba vkládání	-	-	65 s	132 s

Zdroj: vlastní

Legenda k tabulce 2: Výsledky měření:

- ICMP: Průměrná odezva.
- TCP: Propustnost v Mbps.
- UDP: Procentuální ztráta datagramů.
- Doba vkládání: čas jak dlouho trvalo, než se tabulky zaplnily generovanými Flow pravidly.

Naměřená data ukazují, že hardwarové tabulky se využívají maximální potenciál přepínače Aruba. Její nevýhoda je oproti softwarových tabulek velikost. Lze vložit pouze kolem 500 pravidel, nezávisle na jejich velikosti. Další nevýhoda, je delší čas, po kterou se pravidla vkládají do tabulky.

Softwarové tabulky jsou na tom o poznání lépe co se týče velikosti a času vkládání. Do softwarových tabulek, lze vložit až 65 536 Flow pravidel za přibližně stejný čas, jako se do hardwarových tabulek vkládalo pouhých 512. Je to z důvodu, že HW tabulky využívají speciální typ paměti (ASIC), které nefungují jako pouhé „úložiště“, ale jedná se o specifický čip, který dokáže i efektivně vyhledávat a porovnávat příchozí zprávy a pravidla.

Nicméně komunikace za použití SW tabulek je silně omezená. V řádu KB/s, přičemž Aruba poskytuje až 1 Gbps. Z tohoto důvodu nelze doporučit nasadit přepínač Aruba s využitím softwarových tabulek do klasické sítě. Lze jej ale využít pro správu, konfiguraci, či pro nestandardní komunikaci. Například lze nastavit, že pokud přijde zpráva, který neodpovídá žádnému Flow pravidlu uvnitř hardwarové tabulky. Předá se taková zpráva dále v pipeline softwarové tabulce, která bude obsahovat pravidlo pro takovou zprávu. Například by se mohl zaslat na server, který loguje podezřelé připojení.

Co se týká rozdílů naplněných a nenaplněných Flow tabulek, ať softwarových nebo hardwarových, nebyly zjištěny žádné výkyvy v komunikaci. To znamená, že Aruba má implementované efektivní vyhledávání pomocí indexace, hashe, či kombinace obojího.

4.15 Troubleshooting

Přepínač Aruba nabízí debuggovací mód díky, kterému lze snadněji odhalit chybu. Tento mód lze nakonfigurovat pomocí příkazu, který povolí vypisovat debugovací zprávy pro všechny OpenFlow událost, jako přidávání, mazání, či modifikaci protokolu OpenFlow.

```
Aruba-3810M-48G-PoEP-1-slot(config)# debug openflow <errors | events | instance | packets>
```

Tento režim dává možnost vypisovat logy na výstup obrazovky (lze nastavit server, kam se mají odesílat). (HPE ArubaOS-Switch OpenFlow v1.3 Administrator Guide for 16.03, 2017)

Je třeba s tímto nástrojem pracovat opatrně. Pokud se nenakonfigurují správně služby, které mají být sledovány, může nastat situace, že se na konzoli začnou vypisovat veškeré akce, které přepínač Aruba právě provádí. Většinou tento případ končí kompletním zamrznutím přepínače a jediná možnost bývá vynutit tvrdý restart zařízení.

ZÁVĚR

Cílem této práce bylo seznámit čtenáře se základními funkcionalitami protokolu OpenFlow. Jsou zde vysvětleny pojmy jako OpenFlow přepínač, kontrolér, Flow tabulka, či Flow pravidlo. Kromě těchto základních komponent a struktur, je zde také uvedeno jak protokol OpenFlow zpracovává události pomocí OpenFlow pipeline.

Cílem praktické části bylo testování 2 režimů konfigurace protokolu OpenFlow, kdy jsou Flow pravidla ukládána pouze do hardwarových, či do pouze softwarových tabulek. Pro oba typy tabulek je navíc testováno, zda nedochází ke změně chování, pokud jsou tabulky prázdné (pouze s potřebnými pravidly), nebo plné (s generovanými pravidly). Tento cíl byl stanoven z důvodu, že oficiální dokumentace protokolu OpenFlow a společnosti HPE neuvádí rozdíl mezi těmito dvěma způsoby konfigurace, jako propustnost komunikace, latence či dobu vkládání Flow pravidel do tabulek. Správce sítě si musí tedy zjistit tyto charakteristiky sám, což není ideální přístup.

Pro toto testování byl použit přepínač Aruba (3810M) od společnosti HPE. Jedná se o L3 přepínač, který podporuje zpracování komunikace pomocí protokolu OpenFlow. Jako kontrolér byl zvolen Ryu, který byl implementován na virtualizovaném prostředí Linux. Nadále byly využity následující nástroje: protokol ICMP, iPerf3 a směrovač MikroTik, který poskytuje další užitečné nástroje pro testování a měření sítě.

Výsledky jednotlivých testů ukázaly zásadní rozdíly mezi využitím hardwarových a softwarových tabulek. Naměřené hodnoty vypovídají, že komunikace s využitím hardwarových tabulek dosahuje maximálních kapacit, kterých je přepínač Aruba schopný. Nicméně jelikož je pro hardwarové tabulky určena speciální separovaná paměť (ASIC), která je daleko nákladnější na výrobu a je mnohonásobně menší než operační paměť RAM, lze do těchto tabulek vložit mnohonásobně méně záznamů oproti softwarovým tabulkám. Zápis Flow pravidel do těchto tabulek je také o mnoho pomalejší. Tyto nedostatky však vyrovná rychlost komunikace, která tato konfigurace poskytuje.

Mezi hlavní výhody softwarových tabulek je, že se do nich vejde mnohonásobně více Flow pravidel. Oproti hardwarovým tabulkám se do softwarových tabulek ukládají Flow pravidla mnohonásobně rychleji, avšak jejich největší omezení je, že využívají paměť RAM. Paměť RAM je sice mnohonásobně větší, ale zároveň je mnohonásobně pomalejší. Pro vyhledávání v paměti RAM je zapotřebí CPU, na rozdíl od ASIC.

Testy nadále prokázaly, že nezáleží na komplexnosti Flow pravidel. Kapacita softwarových i hardwarových tabulek zůstala neměnná v případech, kdy Flow pravidla obsahovaly pouze zdrojový a cílový port, či se skládaly z několika atributů jako zdrojová a cílová MAC adresa, zdrojová a cílová IPv6 adresa a zdrojový a cílový port.

Z těchto výsledků byl vyvozen závěr, že softwarové tabulky nelze doporučit pro běžný druh komunikace, ale hodí se spíše pro logování, troubleshooting, či konfiguraci sítě. Hardwarové tabulky lze nasadit do běžného provozu, aniž by docházelo k omezení komunikace.

POUŽITÁ LITERATURA

[1] **Open Networking Foundation**. Software-Defined Networking (SDN) Definition. [online]. 2019 [cit. 2019-07-20]. Dostupné z: https://www.opennetworking.org/sdn-definition/

[2] **Open Network Foundation**. OpenFlow Switch Specification. [online]. 2015 [cit. 2019-07-20]. Dostupné z: https://www.opennetworking.org/wp-content/uploads/2014/10/openflow-switch-v1.5.1.pdf

[3] **Hewlett Packard Enterprise**. HPE ArubaOS-Switch OpenFlow v1.3 Administrator Guide for 16.03. [online]. 2017 [cit. 2019-07-20]. Dostupné z: https://support.hpe.com/hpsc/doc/public/display?docId=emr_na-c05365339&docLocale=en_US

[4] **Ryu SDN Framework Community**. Build SDN Agilely. [online]. 2017 [cit. 2019-07-20]. Dostupné z: https://osrg.github.io/ryu/

[5] **SDN Hub**. Ryu Controller Tutorial. [online]. [cit. 2019-07-20]. Dostupné z: http://sdnhub.org/tutorials/ryu/

[6] **Ryu SDN Framework Community.** simple_switch_13. [online]. 2017 [cit. 2019-07-20]. Dostupné z: https://github.com/osrg/ryu/blob/master/ryu/app/simple_switch_13.py

[7] **Hewlett Packard Enterprise Support Center.** Aruba 3810M Switch Series - Specifications. [online]. [cit. 2019-07-20]. Dostupné z: https://support.hpe.com/hpsc/doc/public/display?docId=emr_na-c04940273

[8] **SIA Mikrotīkls**. Products hAP lite. [online]. [cit. 2019-07-20]. Dostupné z: https://mikrotik.com/product/RB941-2nD

[9] **Informit**. Network Layer/Internet Protocols. [online]. 2002 [cit. 2019-07-20]. Dostupné z: http://www.informit.com/articles/article.aspx?p=26557&seqNum=5

[10] **ESnet/LBNL** . iPerf - The ultimate speed test tool for TCP, UDP and SCTP. [online]. [cit. 2019-07-20]. Dostupné z: https://iperf.fr/

[11] **NETSVET**. Malá učebnice OpenFlow (1) – SDN vs. OpenFlow. [online]. 2014 [cit. 2019-07-20]. Dostupné z: https://www.netsvet.cz/mala-ucebnice-openflow-1-sdn-vs-openflow/

PŘÍLOHY

PŘÍLOHA A – SIMPLE_SWITCH_13

Skript SimpleSwitch13 použitý a upravený pro testování.

```
# Copyright (C) 2011 Nippon Telegraph and Telephone Corporation.
#
# Licensed under the Apache License, Version 2.0 (the "License");
# you may not use this file except in compliance with the License.
# You may obtain a copy of the License at
#
#    http://www.apache.org/licenses/LICENSE-2.0
#
# Unless required by applicable law or agreed to in writing, software
# distributed under the License is distributed on an "AS IS" BASIS,
# WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or
# implied.
# See the License for the specific language governing permissions and
# limitations under the License.

from ryu.base import app_manager
from ryu.controller import ofp_event
from ryu.controller.handler import CONFIG_DISPATCHER, MAIN_DISPATCHER
from ryu.controller.handler import set_ev_cls
from ryu.ofproto import ofproto_v1_3
from ryu.ofproto import ether
from ryu.lib.packet import packet
from ryu.lib.packet import ethernet
from random import randint, choice
from string import hexdigits

class SimpleSwitch13(app_manager.RyuApp):
    OFP_VERSIONS = [ofproto_v1_3.OFP_VERSION]

    def __init__(self, *args, **kwargs):
        super(SimpleSwitch13, self).__init__(*args, **kwargs)
```

```
    self.mac_to_port = {}
    self.logger.info("--------------------INIT2----------------")
    self.logger.info(self._random_MAC())
    self.logger.info(self._random_IPv4())
@set_ev_cls(ofp_event.EventOFPSwitchFeatures, CONFIG_DISPATCHER)
def switch_features_handler(self, ev):
    datapath = ev.msg.datapath
    ofproto = datapath.ofproto
    parser = datapath.ofproto_parser

    # install table-miss flow entry
    #
    # We specify NO BUFFER to max_len of the output action due to
    # OVS bug. At this moment, if we specify a lesser number, e.g.,
    # 128, OVS will send Packet-In with invalid buffer_id and
    # truncated packet data. In that case, we cannot output packets
    # correctly.  The bug has been fixed in OVS v2.1.0.
    match = parser.OFPMatch()
    actions = [parser.OFPActionOutput(ofproto.OFPP_CONTROLLER,
                                      ofproto.OFPCML_NO_BUFFER)]
    self.add_flow(datapath, 0, match, actions)

    i = 1
    # 1022 pravidel je pri tomto nastaveni maximum
    while i < 500:
        self._generate_flows(ev)
        i += 1

def add_flow(self, datapath, priority, match, actions, buffer_id=None):
    ofproto = datapath.ofproto
    parser = datapath.ofproto_parser

    inst = [parser.OFPInstructionActions(ofproto.OFPIT_APPLY_ACTIONS,
                                         actions)]
```

```
    if buffer_id:
        mod = parser.OFPFlowMod(datapath=datapath, buffer_id=buffer_id,
                                priority=priority, match=match, table_id=0,
                                instructions=inst)
    else:
        mod = parser.OFPFlowMod(datapath=datapath, priority=priority,
                                match=match, table_id=0, instructions=inst)
    datapath.send_msg(mod)

@set_ev_cls(ofp_event.EventOFPPacketIn, MAIN_DISPATCHER)
def _packet_in_handler(self, ev):
    # If you hit this you might want to increase
    # the "miss_send_length" of your switch
    if ev.msg.msg_len < ev.msg.total_len:
        self.logger.debug("packet truncated: only %s of %s bytes",
                          ev.msg.msg_len, ev.msg.total_len)
    msg = ev.msg
    datapath = msg.datapath
    ofproto = datapath.ofproto
    parser = datapath.ofproto_parser
    in_port = msg.match['in_port']

    pkt = packet.Packet(msg.data)
    eth = pkt.get_protocols(ethernet.ethernet)[0]

    dst = eth.dst
    src = eth.src

    dpid = datapath.id
    self.mac_to_port.setdefault(dpid, {})

    self.logger.info("packet in %s %s %s %s", dpid, src, dst, in_port)

    # learn a mac address to avoid FLOOD next time.
```

```
        self.mac_to_port[dpid][src] = in_port

        if dst in self.mac_to_port[dpid]:
            out_port = self.mac_to_port[dpid][dst]
        else:
            out_port = ofproto.OFPP_FLOOD

        actions = [parser.OFPActionOutput(out_port)]

        # install a flow to avoid packet_in next time
        if out_port != ofproto.OFPP_FLOOD:
            match = parser.OFPMatch(in_port=in_port, eth_dst=dst)
            # verify if we have a valid buffer_id, if yes avoid to send both
            # flow_mod & packet_out
            if msg.buffer_id != ofproto.OFP_NO_BUFFER:
                self.add_flow(datapath, 1, match, actions, msg.buffer_id)
                return
            else:
                self.add_flow(datapath, 1, match, actions)
        data = None
        if msg.buffer_id == ofproto.OFP_NO_BUFFER:
            data = msg.data

        out = parser.OFPPacketOut(datapath=datapath, buffer_id=msg.buffer_id,
                                  in_port=in_port, actions=actions, data=data)
        datapath.send_msg(out)

    def _generate_flows(self, ev):
        msg = ev.msg
        datapath = msg.datapath
        ofproto = datapath.ofproto
        parser = datapath.ofproto_parser
        src = self._random_MAC()
        dst = self._random_MAC()
```

```
            ip4Src = self._random_IPv4()
            ip4Dst = self._random_IPv4()

            ip6Src = self._random_IPv6()
            ip6Dst = self._random_IPv6()

            out_port = ofproto.OFPP_FLOOD

            actions = [parser.OFPActionOutput(out_port)]
            match = parser.OFPMatch(eth_src = src, eth_dst =
dst,eth_type=ether.ETH_TYPE_IPV6, ipv6_src = ip6Src, ipv6_dst = ip6Dst)
            # , ipv4_src = ip4Src, ipv4_dst = ip4Dst, ipv6_src = ip6Src, ipv6_dst = ip6Dst
tcp_src = randint(1000, 2000), tcp_dst = randint(1000, 2000)
            self.add_flow(datapath, 100, match, actions)

            #match2 = parser.OFPMatch(eth_type=0x0800, #ip_proto=6, tcp_src = 1400, tcp_dst
= 1450)
            #self.add_flow(datapath, 100, match2, actions)

        def _random_MAC(self):
            return "52:54:00:%02x:%02x:%02x" % (
                randint(0x00, 0x7f),
                randint(0x00, 0xff),
                randint(0x00, 0xff))

        def _random_IPv4(self):
            octets = []
            for x in range(4):
                octets.append(str(randint(0,255)))
            return '.'.join(octets)

        def _random_IPv6(self):
            octets = []
```

```
for x in range(8):
    octet = []
    for x in range(4):
        octet.append(str(choice(hexdigits.lower())))
    octets.append(''.join(octet))
return ':'.join(octets)
```

PŘÍLOHA B – KONFIGURACE ARUBA

Konfigurační skript pro přepínač Aruba

```
hostname "Aruba-3810M-48G-PoEP-1-slot"
module 1 type jl074x
module 2 type jl074y
snmp-server community "public" unrestricted
openflow
   controller-id 5 ip 192.168.10.10 controller-interface vlan 2
   instance "dipp"
      exit
   instance "dipprace"
      listen-port
      member vlan 3
      controller-id 5
      version 1.3 only
      max-backoff-interval 3
      table-num policy-table 0
      table-num sw-table-1 1
      exit
   instance "dippprace"
      exit
   enable
   exit
oobm
   ip address dhcp-bootp
   exit
vlan 1
   name "DEFAULT_VLAN"
   no untagged 1,3-5
   untagged 2,6-48
   ip address dhcp-bootp
   exit
vlan 2
```

```
  name "VLAN2"
  untagged 1
  ip address 192.168.10.11 255.255.255.0
  exit
vlan 3
  name "VLAN3"
  untagged 3-5
  ip address 192.168.0.10 255.255.255.0
  exit
```