VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

Fakulta elektrotechniky
a komunikačních technologií

BAKALÁŘSKÁ PRÁCE

Brno, 2020 Daniel Kříž

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ
FACULTY OF ELECTRICAL ENGINEERING AND COMMUNICATION

ÚSTAV TELEKOMUNIKACÍ
DEPARTMENT OF TELECOMMUNICATIONS

SDN PRO ŘÍZENÍ SÍTÍ LAN
SDN IN LAN NETWORKS

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE — **Daniel Kříž**
AUTHOR

VEDOUCÍ PRÁCE — **Mgr. Karel Slavíček, Ph.D.**
SUPERVISOR

BRNO 2020

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ

Bakalářská práce

bakalářský studijní program **Telekomunikační a informační systémy**

Ústav telekomunikací

Student: Daniel Kříž ***ID:*** 203269
Ročník: 3 ***Akademický rok:*** 2019/20

NÁZEV TÉMATU:

SDN pro řízení sítí LAN

POKYNY PRO VYPRACOVÁNÍ:

Cílem práce je analyzovat současné možnosti využití technologie SDN v oblasti sítí LAN a vytipovat k tomuto účelu vhodný SDN kontroler. Věcným výstupem práce je implementace tohoto kontroleru na jednom či více virtuálních serverech a vytvoření laboratorní úlohy, na které se budou moci další studenti seznámit s technologií SDN a jejím využití pro sítě LAN.

DOPORUČENÁ LITERATURA:

[1] SDN: Software Defined NEtworks [Book]. O'Reilly Media - Technology and Business Training [online] Copyright (c) 2019 Safari Books Online. [cit. 15.09.2019]. Dostupné z: https://www.oreilly.com/library/view/sdn-software-defined/9781449342425/

[2] GORANSSON, Paul, Cuck BLACK a Timothy CULVER, Software defined networks: a comprehensive approach. Second edition. Singapore: Morgan Kaufmann, [2017]. ISBN 9780128045558.

Termín zadání: 3.2.2020 ***Termín odevzdání:*** 8.6.2020

Vedoucí práce: Mgr. Karel Slavíček, Ph.D.

prof. Ing. Jiří Mišurec, CSc.
předseda rady studijního programu

ABSTRAKT

Tato bakalářská práce je věnována problematice softwarově definovaných sítí. V práci je popsán koncept SDN, protokol OpenFlow, který je nedílnou součástí této problematiky a v poslední řadě je zde zmíněn koncept NFV. Cílem je udělat si přehled informací o této problematice, projít dostupné open source kontroléry a vybrat z nich nejvhodnější. Dále pak vytvořit návrh realizace laboratorní úlohy a vytvořit seznam dostupných komponent. Závěrečným úkolem je sestrojit a otestovat laboratorní úlohu a následně za pomocí těchto znalostí vytvořit laboratorní návod.

KLÍČOVÁ SLOVA

SDN, OpenFlow, virtualizace, NETCONF, YANG, LAN, Ryu, Softwarově definované sítě

ABSTRACT

This bachelor's thesis is devoted to the problem of Software Defined Networks. It describes the concept of SDN, the OpenFlow protocol, which is an integral part of this issue, and lastly is mentioned the concept of NFV. The goal is to research this issue, gather information about the available open source controllers and choose the most suitable ones. Furthermore, a proposal for the implementation of the laboratory task and a list of available components have to be created. The final task is to construct and test laboratory task and then with help of this knowledge create laboratory manual.

KEYWORDS

SDN, OpenFlow, virtualization, NETCONF, YANG, LAN, Ryu, Software defined networking

KŘÍŽ, Daniel. *SDN pro řízení sítí LAN*. Brno, Rok, 103 s. Bakalářská práce. Vysoké učení technické v Brně, Fakulta elektrotechniky a komunikačních technologií, Ústav telekomunikací. Vedoucí práce: Mgr. Karel Slavíček, Ph.D.

PROHLÁŠENÍ

Prohlašuji, že svou bakalářskou práci na téma „SDN pro řízení sítí LAN“ jsem vypracoval samostatně pod vedením vedoucího bakalářské práce a s použitím odborné literatury a dalších informačních zdrojů, které jsou všechny citovány v práci a uvedeny v seznamu literatury na konci práce.

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

Brno ....

podpis autora

PODĚKOVÁNÍ

Rád bych poděkoval vedoucímu bakalářské práce panu Mgr. Karlovi Slavíčkovi, Ph.D. za odborné vedení, konzultace, trpělivost a podnětné návrhy k práci.

Obsah

Seznam obrázků

Seznam tabulek

Úvod

Z počátku vývoje síťových technologií byly sítě jednoduché, ale jak se začala technologie vyvíjet, zvyšoval se počet síťových zařízení, která byla technologicky vyspělejší. To vedlo k obtížnější a složitější správě sítě. Sítě obsahovaly spoustu druhů zařízení od rozbočovačů přes přepínače a směrovače až po firewally, nebo překladače adres. Tato zařízení přicházela stále více komplexní a poskytovaný software byl většinou uzavřený a proprietární. S příchodem nových zařízení a softwaru přišly i nové protokoly, které podléhaly standardizaci.

S nástupem nových technologií začali přibývat i noví výrobci. Každý z těchto výrobců disponoval ve svých zařízeních originálním konfiguračním rozhraním a operačním systémem. Konfigurování a správa sítě se stávala obtížnější. S tímto vývojem sice přišly i nástroje, které umožňovaly správu sítě, ale vysoká cena je činila nedostupnými. Díky tomu se zvýšila složitost a náklady na provoz sítě.

S příchodem SDN přišla inovace v návrhu a správě sítě. Jednou inovací oproti tradičním sítím je rozdělení datové a řídící roviny. Druhou inovací je sjednocení řídících částí, čímž je umožněno ovládání více datových částí. Tato technologie poskytuje nástroj, tzv. kontrolér, který slouží jako mozek celé sítě, umožňuje celkovou správu sítě a konfiguraci více zařízení nezávisle na jejich konfiguračním rozhraní.

Úkolem této práce má být sestrojení laboratorní úlohy, která má ostatním ukázat a vysvětlit princip využití SDN v lokálních sítích. Další možností použití SDN je například ve WAN, kde je tato architektura představena pod názvem SD-WAN. Zde je cílem spravovat větší celky sítí. SDN lze uplatnit i v jiných oblastech a to například v oblasti cloud computing. [5]

1 Úvod do virtualizace sítí

Obecně pojem virtualizace popisuje proces vytváření virtuálních instancí na jedné hardwarové platformě za využití jednotlivých prvků této platformy, například: procesor, paměť, úložiště a další. Každá tato instance se chová jakoby byla osamostatněná a měla vlastní komponenty, čímž je umožněno efektivnější využití hardwaru. Na jedné hardwarové platformě může být současně více virtuálních instancí, které lze libovolně ovládat a měnit jejich parametry. Díky výše zmíněným informacím dochází ke zvýšení efektivity daného zařízení (PC) a zároveň ke snížení finančních prostředků na realizaci. [1] [2]

1.1 Princip

Princip virtualizace spočívá v oddělení fyzických zdrojů a virtuálního prostředí za pomoci hypervizoru. Jedná se o program, který se stará o správu jednotlivých virtuálních strojů. Zároveň tento program funguje jako rozhraní mezi fyzickým a virtuálním hardwarem, poskytuje virtuálním strojům přístup k fyzickým zdrojům a rozděluje je podle potřeby. Virtuální stroj se chová jako datový soubor, který lze přesouvat z počítače na počítač. [3]

1.1.1 Typy hypervizoru

Hypervizor se rozděluje se na dva typy:

- Bare-metal - slouží jako náhrada za operační systém a spolupracuje s hardwarovými zdroji.
- Hosted - představuje aplikaci Operačního systému, která se využívá na koncových zařízeních a slouží k řízení základních hardwarových prostředků. [1] [4]

1.2 Výhody

Virtualizace v oblasti IT zajišťuje značné výhody a možnosti. Jednou z jejích výhod je zvýšení efektivity zdrojů. Před příchodem této technologie musel mít každý server svůj vlastní hardware a vyhrazené místo, kde se nacházela ve většině případů pouze jedna aplikace. Na každém z těchto serverů musela být samostatně provedena konfigurace. Ve výsledku pak byla využita jen část serverů a zbytek jen zabíral prostor. Virtualizace serverů umožňuje mít jeden hardware, na kterém může běžet více virtuálních strojů. Každý z nich může být určen pro jednotlivé aplikace, čímž je zaručena úspora výkonu. [4]

Druhou výhodou je umožnění snadnější správy. Nahrazením fyzických zařízení virtuálními se značně zjednodušila správa zásad napsaných v softwaru. Došlo k automatizaci některých procesů, například: tvoření kolekcí virtuálních strojů, čímž bylo umožněno instalovat služby efektivně a bez zdlouhavého zpoždění.

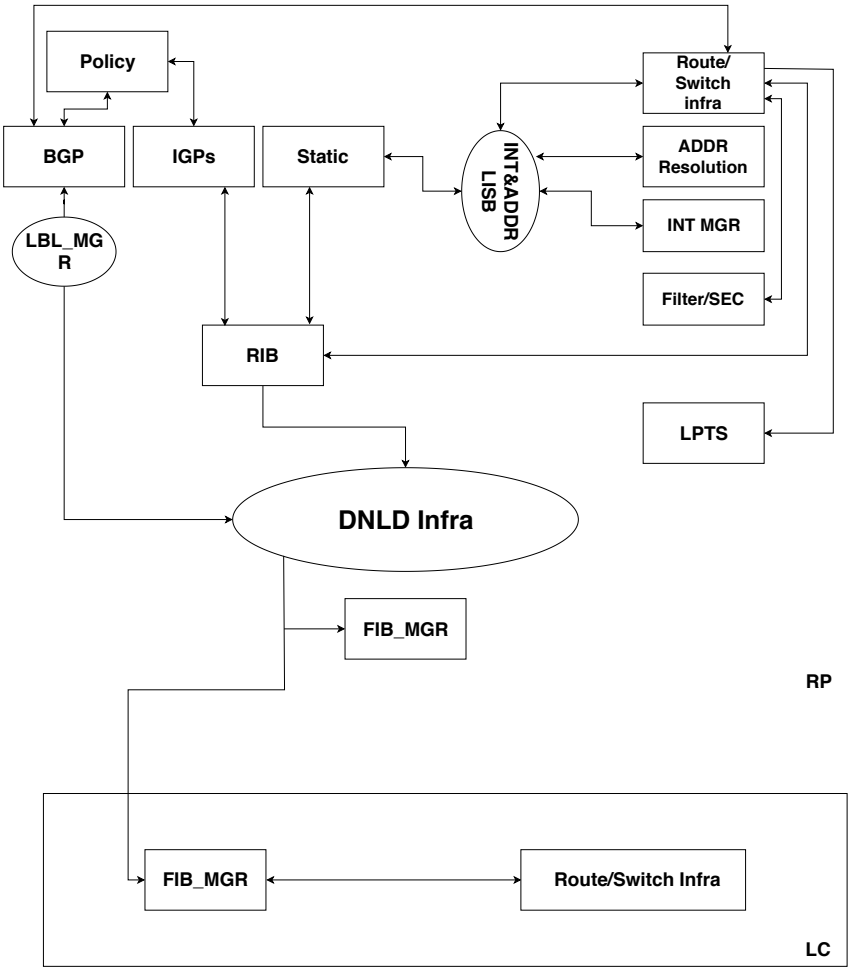
Mezi poslední výhodu patří zkrácení čekací doby. Virtualizace, v případě pádu serveru, umožňuje nahradit daný virtuální server jiným a tím přebrat službu, kterou provozoval. [4]

2 Softwarově definované sítě

SDN je architektura, která poskytuje dynamičnost, snadnější správu a větší cenovou dostupnost dnešních sítí. Základní vlastností je oddělení řídící a datové roviny, čímž usnadňuje práci síťovým zařízením. Centrálním mozkem této sítě je kontrolér, který odesílá instrukce přepínačům a tím řídí celý její chod. Základním komunikačním protokolem je OpenFlow, jenž umožňuje komunikaci s nižšími vrstvami architektury. Tato architektura je podrobněji popsána níže. [6]

2.1 Architektura SDN

Architektura tohoto typu sítě je rozdělena na dvě roviny, řídící a datovou, které se nacházejí v oddělených zařízeních. Tuto problematiku můžeme popsat schématicky takto: [6]



Obr. 2.1: Kontrolní a datový plán zařízení [6]

Na obrázku 2.1 je popsána liniová karta (LC), kde je umístěna datová rovina a trasa procesoru (route processor, RP box), ve které je umístěna řídící rovina. Dále jsou zde zobrazeny základní služby (BGP, IGPs, atd.), jenž slouží k ověřování/oznamování linkové dostupnosti, kvality informací, objevování sousedů a rozlišování adres. Dále je zde zobrazeno řízení RIB-to-FIB (popsáno níže které je srdcem řídící roviny. Jelikož tyto služby mají velmi těsné výkonové smyčky (pro krátké doby detekce událostí), jsou téměř vždy, nezávisle na strategii řídící roviny, součástí datové roviny. [6]

2.1.1 Řídící rovina

Stará se o logiku sítě a řízení toku dat. Jednou z funkcí této roviny je ukládání směrovacích tabulek a síťové topologie. Tyto informace poskytuje ostatním zařízením v síti. Dále se stará o směrování dat v síti, komunikaci mezi jednotlivými řídícími částmi v síti, sledování provozu, signalizování stavů, výměnu dat a konfiguraci. Tyto informace jsou poté předávány datové rovině, která je využívá ke směrování provozu mezi vstupními a výstupními porty zařízení. SDN si udržuje tzv. směrovací informační základnu (RIB), v níž je uložena topologie sítě. RIB je udržována konzistentní, a to prostřednictvím výměny dat mezi jednotlivými řídícími částmi v síti.

Dále má k dispozici tzv. předávací informační základnu (FIB), která obsahuje směrovací údaje. FIB se objevuje mezi řídící a datovou rovinou. Pokud je RIB považována za konzistentní a stabilní, provede se naprogramování FIB. Nejdříve však musí řídící jednotka vytvořit pohled na topologii sítě, která splňuje předem nadefinovaná kritéria. Tento pohled je buď naprogramován ručně, nebo je zprostředkován směrovacím protokolem (OSPF, RIP, atd.). Veškeré zmíněné procesy řídí kontrolér. [6]

2.1.2 Datová rovina

Stará se o zpracování příchozích datagramů a předávání z portu na port díky informacím od řídící roviny. Prostřednictvím operací spojové vrstvy dochází k základní kontrole datagramu. U správně vytvořeného datagramu je provedeno vyhledávání v tabulce FIB, kterou již předtím naprogramovala řídící rovina. Tomuto se někdy říká rychlá cesta pro zpracování paketu, protože k identifikaci je zapotřebí pouze přeprogramovaná FIB tabulka. Výjimkou je situace, kdy je detekovaný neznámý cíl paketu. Tyto pakety jsou odeslány řídícímu procesoru, kde jsou řídící rovinou zpracovávány pomocí RIB. Ke zpracovávání FIB tabulek se využívá specializovaný hardware ASIC.

Datová rovina může implementovat některé další služby a funkce, které jsou využitelné ke směrování, např. ACL (Access Control List) nebo QoS (Quality of

Services). Díky těmto funkcím lze (do malé míry) změnit, nebo vyloučit výsledek vyhledávání.

Z těchto poznatků vyplývá, že řídící rovina je osamostatněna v zařízení zvaném řídící kontrolér. Datová rovina se nachází stále v síťovém zařízení nazývaném přepínač. Soustava těchto dvou rovin tvoří ucelenou topologii. [6]

2.1.3 Přesun informací mezi rovinami

Protože v reálném více-slotovém (distribuovaném) systému je řídící a datová rovina oddělena každá na jiné linkové kartě, existuje komunikace, která i v tomto případě poskytuje odolnost vůči chybám. Nejprve je třeba se zaměřit na koncept základního předávání paketů. Nabízí se otázka jestli je zde možnost, že by provoz distribuované sítě byl přerušen z důvodu dezinformace datové roviny. Aby k tomuto jevu nedocházelo musí být datová rovina synchronizovaná s jednotlivými elementy řídící sítě. Proto se využívají systémy detekce distribučních chyb pro předávání tabulek, které mohou být zabudovány v jednotlivých datech (verzování tabulky), nebo v přenosovém mechanismu (podepsáním tabulky hashem, nebo cookie).

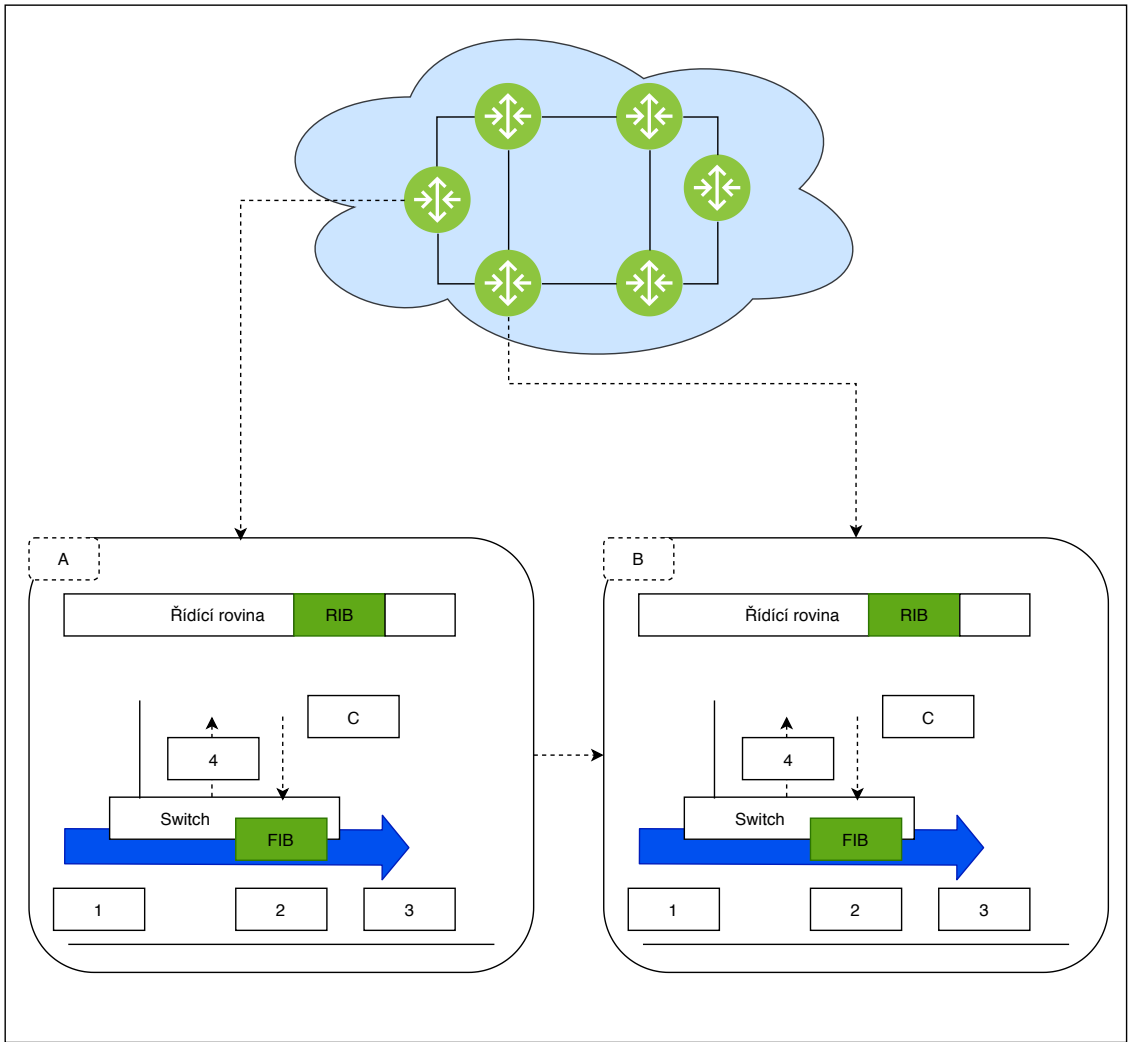
Někteří výrobci implementovali hardwarové vylepšení, které poté co řídící rovina naprogramuje datovou, zkontroluje jednoduché chyby v předávacím čipu a doplňkové paměti. Dále je zde možnost kontrolování i celých chybových bloků pomocí dalších vylepšení. Ta jsou však drahá a implementují se spíše na pozadí.

Některé takovéto systémy využívají dvoustupňové vyhledávání. V první fázi proběhne identifikování výstupního slotu a následně se provede druhý stupeň vyhledávání. Díky tomu dochází ke zmenšení výstupních FIB, čímž dojde ke zrychlení sítě. Mohou se ale vyskytnout obtížně detekovatelné dvoustupňové asynchronní ztráty, které mohou provoz přerušit. [6]

2.1.4 Princip komunikace datové a řídící roviny

Na obrázku 2.2 je znázorněn princip komunikace mezi rovinami. V horní části je zobrazena síť přepínačů, které jsou vzájemně propojeny. Níže se nachází detail řídící a datové roviny dvou přepínačů (uzly A a B), kde každá z těchto rovin je umístěna na vlastním procesoru/kartě, ale obě jsou umístěny ve stejném zařízení. Obrázek zobrazuje předávání paketů z přepínače A na přepínač B. Na obrázku je zobrazen příklad, kdy jsou pakety přijímány vstupními porty linkové karty, kde sídlí datová rovina.

V případě, že paket přijde z neznámé MAC adresy, nebo se jedná o změnu v řízení provozu (pakety obsahující LSA), dojde k potrestání (plunted), nebo přesměrování (4) paketu na řídící rovinu zařízení. Jakmile paket dorazí, informace o něm jsou

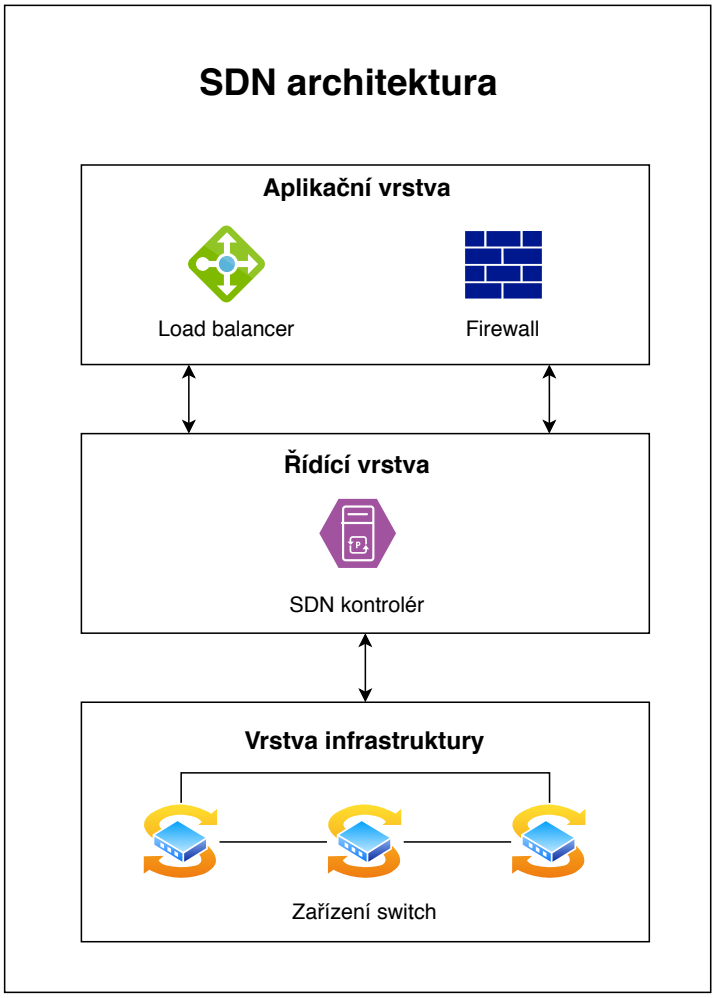


Obr. 2.2: Kontrolní a datová rovina typické sítě [6]

zpracovány, případně dojde ke změně RIB a s tím související odeslání aktualizačních zpráv ostatním uzlům. Když se stane RIB stabilní, aktualizuje se FIB na datové i řídící rovině. V našem případě se jedná o paket s neznámou zdrojovou MAC adresou, proto řídící rovina vrátí paket (C) na datovou rovinu (2), kde dojde k předání paketu (3). Pokud je nutné další přeprogramování FIB, provede se také v kroku (C), což je náš případ (naučení zdrojové MAC adresy). Stejný proces se provede i na přepínači (B). [6]

2.1.5 Vrstvy SDN

Architekturu SDN můžeme také rozdělit na 3 vrstvy, kde vždy sousední vrstvy spol mohou komunikovat. Nazývají se aplikační, řídící a vrstva infrastruktury, jak je ostatně ukázáno na následujícím schématu.



Obr. 2.3: Architektura SDN [22]

Aplikační vrstva

Vrstva, která se nachází na nejvyšší pozici v SDN architektuře. Na této vrstvě se nachází aplikace, které si s kontrolérem předávají informace. Například chování sítě a další potřebné prostředky skrz aplikačně programovatelné rozhraní (API), a to přes tzv. „southbound interface“. Další vlastností této vrstvy je, že dokáží vytvořit abstraktní náhled na síťovou topologii díky informacím, které dostanou od kontroléru, což napomáhá k rozhodování.

Dále na této vrstvě můžeme najít aplikace určené pro management, analýzu nebo podnikové nástroje. Tyto aplikace se velice často využívají ve větších data centrech. Příkladem je analytická aplikace, kterou lze využít k rozpoznávání podezřelé síťové aktivity, anebo pro účely zabezpečení. Díky výše zmíněným vlastnostem může zajišťovat kvalitu služeb, rozložení zátěže, konfiguraci prvků sítě a kontroléru, stanovení pravidel ACL (Access Control List) atd. [7] [8]

Řídící vrstva

Je tvořena kontrolérem, jenž slouží jako mozek celého SDN. Je to logická jednotka dostávající instrukce, nebo požadavky od aplikací, které jsou na aplikační vrstvě. Dále kontrolér disponuje logikou pro ovládání síťových zařízení a provozu, jako je přepínání a směrování paketů, L2 a L3 VPN, zabezpečení branou firewall, DNS, DHCP a klastrování. Jakmile dojde k implementaci těchto služeb, jejich API je vystavené na aplikační vrstvě, což usnadňuje správcům sítě monitorováni, správu a konfiguraci sítě.

Další funkcí této vrstvy je získávání informací ze síťových zařízení datové vrstvy a jejich odesílání zpět aplikační vrstvě spolu s náhledem na topologii sítě a statistikami o síťovém provozu a událostech, které se v síti staly. Dále jsou na této vrstvě shromažďované důležité informace jako jsou stavové údaje, topologické detaily, statistické údaje, atd. Ty jsou neustále aktualizovány a ukládány. Řídící vrstva leží uprostřed celé architektury a disponuje dvěma typy rozhraní – severní a jižní. [7] [8]

Severní rozhraní (Northbound interface)

Toto rozhraní je určeno ke komunikaci s aplikační vrstvou a je obecně realizováno prostřednictvím REST API SDN kontrolérů. Poskytuje následující služby:

- V průběhu komunikace získává a zaznamenává od kontrolérů a ze síťové infrastruktury provozní informace typu QoS, topologie, zpoždění, jitter, zpracovávání datového toku atd. Tyto informace poskytuje aplikační vrstvě.
- Ukládá a následně poskytuje informace o správě síťové infrastruktury, například napájecí zdroje, využití energie, údržba a další stavové informace.
- Poskytuje informace ohledně síťové politiky a pravidel. Například řízení přístupu a zabezpečení atd. [8] [9]

Jižní rozhraní (Southbound interface)

Toto rozhraní se nachází mezi řídící a datovou rovinou. Poskytuje následující služby:

- Poskytuje flexibilitu řídící rovině a umožňuje přizpůsobení se novým změnám v řízení sítě. Využívá se k vytváření a konfiguraci virtuálních sítí.
- Vytváří abstrakci nad fyzickým zařízením a umožňuje aplikacím k ní přistupovat.
- Zajišťuje bezpečnou izolaci mezi několika virtuálními sítěmi.
- Stará se o zapisování informací o fyzických síťových prostředcích. [9]

Slouží ke komunikaci s vrstvou infrastruktury a nacházejí se zde protokoly jako OpenFlow, NETCONF, OVSDB atd. [8]

Vrstva infrastruktury

Tuto vrstvu tvoří síťová zařízení jako například přepínače a směrovače. Předávají, směrují a zpracovávají data v síti. Tato vrstva je považována za fyzickou. Zařízení této vrstvy spolu komunikují skrz jižní rozhraní. [8]

2.2 Módy a jejich vlastnosti a funkce

V SDN jsou možnosti budování sítě rozdělené do tří módů. Mezi ně patří:

2.2.1 Reaktivní mód

V případě, že přijde datový tok do přepínače, OpenFlow agent interaguje s centralizovaným kontrolérem a převádí příkazy do specifických akcí. Současně provede vyhledávání v tabulkách toků (viz. dále) buď ve vyhledávacím ASIC, pokud se nachází v hardwaru, anebo v softwarové tabulce v případě, že se jedná o vSwitch. Jedná se o softwarovou aplikaci, která umožňuje komunikaci mezi virtuálními stroji.

Pokud se nepodaří najít záznam v tabulce, přepínač odešle žádost o další instrukce kontroléru prostřednictví OFP packet-in. Tento mód reaguje na síťový provoz. Dále dojde ke konzultaci s OpenFlow kontrolérem a na základě instrukcí dojde k vytvoření pravidla v tabulce toků.

Hlavním problémem tohoto módu je, že ho nelze použít ve velkých sítích, protože pří velkém počtu síťových zařízení dochází k velkému zatěžování kontroléru, což může vézt až k jeho přetížení. Využití tohoto módu je vhodné v menších sítích o velikosti v řádech maximálně desítek zařízení, nebo v případě potřeby úplné kontroly nad sítí. [10] [11]

2.2.2 Pro-aktivní mód

Základem fungování tohoto módu je nereagovat na příchozí paket, ale průběžně zaplňovat tabulky toků pro všechny možnosti, které mohou vstoupit do přepínače. Průběžným definováním tabulek toků dochází k tomu, že není třeba odesílat packet-in, protože tabulky toků budou aktualizovány při příchodu, čímž je umožněn plynulý průchod dat. Příchozí toky ihned zpracovává TCAM (speciální vysokorychlostní paměť, která prohledává celý svůj obsah v jediném hodinovém cyklu). Díky tomu je docíleno snížení latence, kterou způsobovala komunikace přepínače s kontrolérem pokaždé když přišel nový datový tok. [10]

2.2.3 Hybridní mód

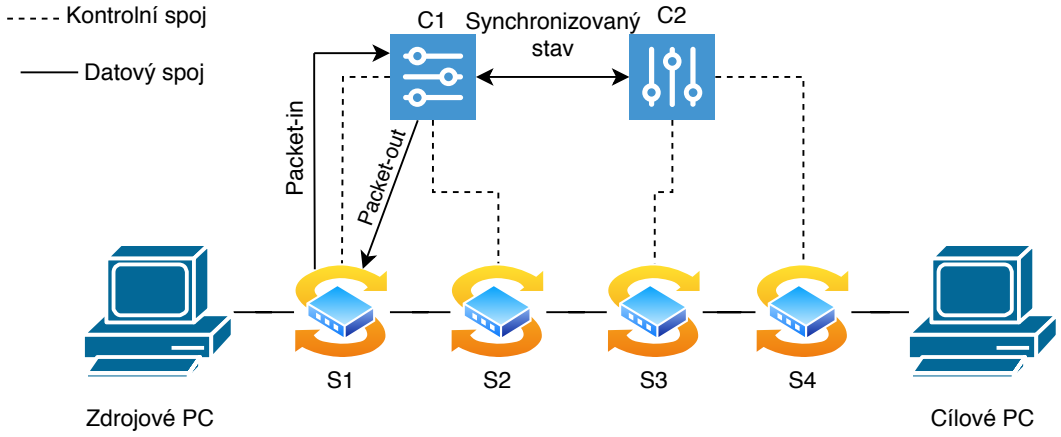
Jedná se o současné využití jak reaktivního tak pro-aktivního módu, čímž je dosaženo výborné škálovatelnosti bez změny chování sítě. Využívá se v sítích, kde je vyžadována jak flexibilita řízení toku dat, tak i zachování nízkého zpoždění.

Ve větších podnicích je kladen důraz na bezpečnost a nízké zpoždění. Obojí dokáže SDN poskytnout. Mimo této kombinace metod lze také využít lokální skupinu přepínačů. Díky tomu je možné k běžným funkcím těchto zařízení přidat i další funkce SDN, například HP Network Protector, který se využívá jako detektor malware. [10]

2.3 Komunikace s více kontrolery

Z důvodu stále se zvětšujících sítí proudí do kontroléru větší množství dat a ten potom nestíhá vyřizovat žádosti jednotlivých zařízení, což zapříčiňuje zahlcení sítě. Proto byl navržen koncept master and slave, jenž umožňuje propojení centrálního kontroléru s ostatními kontroléry.

Na obrázku níže je zobrazena topologie sítě s dvěma kontroléry. Každý z nich se stará o určitou část sítě. Kontroléry (c1) a (c2) si mezi sebou sdílí informace a zároveň je mezi sebou zálohují, takže v případě příchodu paketu na přepínač (s1) mohou být směrovací informace předány všem odpovídajícím přepínačům jedním z kontrolérů. Tím se sníží zahlcení kontroléru a zvýší výkonnost celé sítě. [19]



Obr. 2.4: Komunikace mezi více kontroléry [19]

2.4 Bezpečnost SDN

V neposlední řadě lze SDN využít ke zvýšení zabezpečení sítě. Prostřednictvím Analýz a metod detekce jsou generována bezpečnostní data. Tato data jsou shromažďována centrálním kontrolérem. Analýzou těchto dat lze zlepšit, nebo zaktualizovat bezpečnostní politiku a následně pomocí kontroléru a pravidel toku ji rozšířit napříč sítí. [20]

V současné době se SDN potýká s bezpečnostními problémy. Jedním z nich jsou DoS útoky, které jsou na základě testování snadno proveditelné. Důvodem je volitelné nastavení TLS (kryptografický protokol, který poskytuje zabezpečení end-to-end komunikace) se vzájemnou autentizací mezi kontroléry a přepínači. V případě, že TLS nemá správné nastavení, může útočník za pomoci DoS útoku manipulovat s pravidly architektury SDN.

Jednou z dalších potenciálních hrozeb je úprava dat. Jelikož kontrolér umožňuje programovat síťová zařízení a ovládat datové toky, existuje možnost, že útočník převezme kontrolu nad kontrolérem a je tudíž schopen provádět změny v síťových zařízeních.

Dalším bezpečnostním rizikem je špatná konfigurace SDN. Toto riziko vzniká tím, že obslužné a zabezpečující protokoly se stále vyvíjí a v případě špatného nastavení z důvodu neznalosti se bezpečnost snižuje, čímž se zvyšuje úspěšnost útoků.

Existuje mnoho možností jak vylepšit zabezpečení sítě SDN, a to na každé vrstvě. Jedním z řešení je využít monitorovací systémy k detekci možných hrozeb. Další možností je využití zašifrované komunikace či různých podpůrných bezpečnostních aplikací. Existuje velká škála bezpečnostních hrozeb a jejich řešení. Obecně je zabezpečení sítě SDN složité a není předmětem téhle práce. [21]

2.5 Výhody SDN

Softwarově definované sítě nabízejí spoustu výhod oproti tradičním sítím. Jednou z nich je centralizované řízení více síťových prvků, které podporují SDN. To umožňuje snadnou kontrolu nad síťovým provozem. Tohoto se využívá zejména při zabezpečení, kdy je potřeba kontrolovat datový provoz v síti.

Další výhodou je oddělení řídící a datové roviny, což snižuje zátěž jednotlivých zařízení. Určité činnosti jsou tedy přesunuty na řídící vrstvu, jako například ukládání směrovacích tabulek, řízení provozu, ukládání topologie sítě atd. Kontrolér se tímto stará o chod celé sítě a síťová zařízení slouží pouze k přeposílání paketů. SDN umožňuje jednoduché nasazení nových protokolů a síťových služeb jako výsledek operativní abstrakce. [22] [23]

2.6 Nevýhody SDN

SDN má ale také své nevýhody. Nabízená flexibilita a funkčnost, kterou poskytuje, vyžaduje dodatečný výkon na zařízeních, čímž je ovlivněna rychlost zpracování a propustnost. Na druhou stranu lze díky flexibilitě SDN některé úkoly zjednodušit a tím zmenšit požadavek na potřebný výkon. [22]

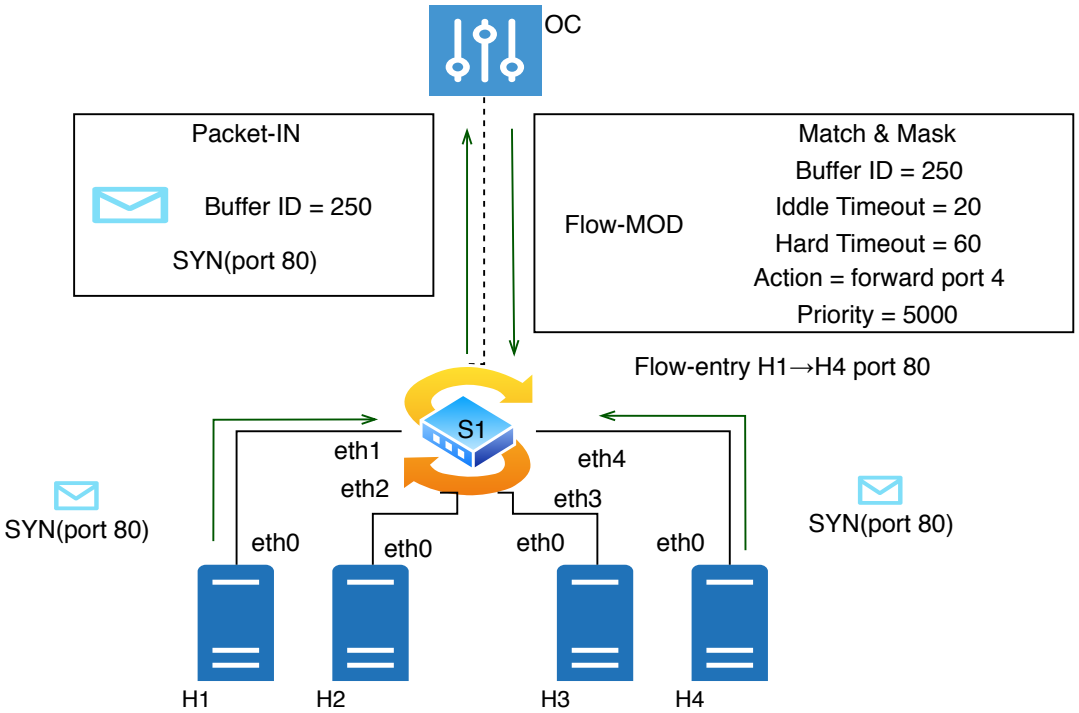
Druhá nevýhoda potom spočívá v použití kontroléru, při jehož napadení je možné snadno převzít kontrolu nad celou sítí. [20]

3 Protokol OpenFlow

OpenFlow je jedním z důležitých neproprietárních protokolů SDN. Jedná se o nástroj, který definuje komunikaci mezi kontrolérem a přepínačem. Komunikace je umožněna skrze zabezpečený kanál. K zabezpečení tohoto kanálu je ve většině případů využito TLS (Transport layer security) asymetrické kryptografie. K této komunikaci využívá sadu oboustranně odlišných zpráv, které si mezi sebou vzájemně posílají. Tyto zprávy umožňují kontroléru naprogramovat přepínač, aby byla umožněna co nejpřesnější kontrola a případná úprava síťového toku.

Protokol OpenFlow se dělí na dvě části wire protocol a konfigurační protokol. První částí je wire protokol (aktuálně verze 1.3.x), který vytváří řídící relace, definuje struktury zprávy pro modifikace toků či sbírání statistik nebo definuje základní struktury přepínače, jako jsou porty a tabulky. Druhá část protokolu, označovaná jako konfigurační a správní protokol, k přiřazování fyzických portů přepínače k určitému kontroléru využívá NETCONF. Dále nastavuje vysokou dostupnost a způsob chování při přerušení spojení s kontrolérem. [6] [24]

3.1 Komunikace v OpenFlow



Obr. 3.1: Komunikace OpenFlow [25]

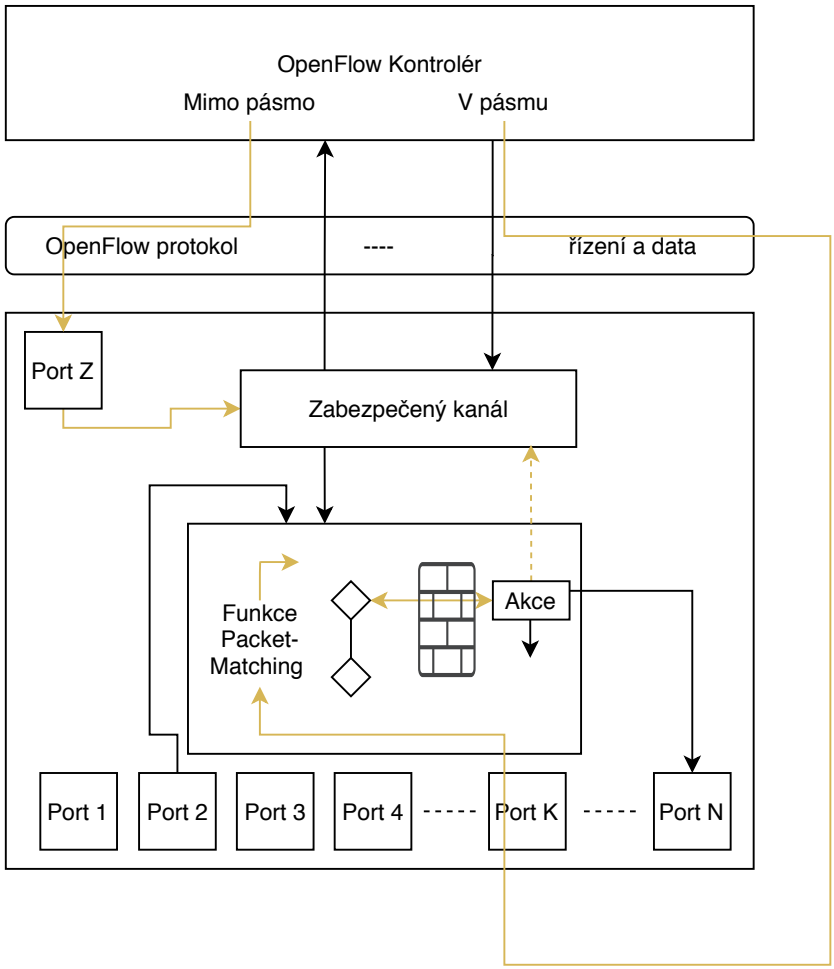
Na obrázku 3.1 je zobrazena topologie, která je využita pro popis komunikace v OpenFlow. Jako příklad je zobrazena komunikace H1 s H4, kde je umístěn http server. Jelikož se jedná o TCP komunikaci, H1 prvně odešle SYN zprávu na přepínač S1 (port 80). Přepínač poté zkontroluje lokální tabulky toků a v případě, že nenajde shodu, přepošle tento paket kontroléru formou zprávy packet-in. Tento paket může obsahovat celou SYN zprávu, nebo hlavičky a Buffer ID. V našem případě 250 (referenční ID vyrovnávací paměti), díky čemuž může kontrolér zjednodušeně říct přepínači co má udělat s uloženým paketem. Po obdržení packet-in může kontrolér odeslat packet-out (viz. 3.3.1), nebo zprávu Flow-MOD přepínači (viz. 3.3.1).

V našem případě dojde k odeslání zprávy Flow-MOD s Buffer ID 250, čímž informuje přepínač o tom, že má vytvořit nový vstup v tabulce toků. Díky tomu bude vědět co dělat, když přijde podobný paket. Touto zprávou dále kontrolér chce říct, aby všechny příchozí žádosti se zdrojovou IP, MAC H1 a cílovou adresou IP, MAC H4 (port 80) odeslal rozhraním 4. Jako další přepínač S1 odešle zprávu SYN na H4. Ten odešle odpověď SYN/ACK, čímž potvrdí přijetí zprávy a celá komunikace se opakuje ve směru H4 k H1. S1 přijme paket a kontroléru odešle packet-in, protože nemá uložený záznam pro tento paket. Kontrolér zpětně odešle packet-out, nebo zprávu Flow-MOD. V případě odeslání Flow-MOD na přepínači S1 dojde k přidání nového záznamu, kde bude zaznamenáno, že veškerá komunikace s H4 a H1 bude směřovat přes rozhraní 1. H1 přijme paket a odešle zpět SYN/ACK. Další komunikace mezi H1 a H4 probíhá bez zásahu kontroléru, protože přepínač S1 má vytvořené záznamy v tabulce toků a již ho není třeba využívat. Závěrem si H1 a H4 vymění zprávy HTTP Request a HTTP Reply bez použití kontroléru. [25]

3.2 Zabezpečený kanál

Jedná se o zabezpečenou komunikační trasu mezi OpenFlow kontrolérem a zařízením. K zabezpečení se využívá asymetrické šifrování založené na protokolu TLS, ale lze využít i pouze TCP spojení bez zabezpečení. Obě připojení lze realizovat uvnitř pásma, nebo mimo pásmo. TCP spojení lze použít v případě, že přepínače jsou umístěny v kontrolovaném prostředí, například datové centrum. Tím dochází k ušetření výkonu.

Na obr. 3.2 je zakreslen mechanismus zabezpečeného kanálu. V případě, že jde o připojení mimo pásmo probíhá komunikace skrz port, který není přepínán datovou rovinou (port Z). Toto připojení můžeme realizovat pouze pokud se jedná o hybridní OpenFlow přepínač, protože OpenFlow zprávy procházející skrze zabezpečený kanál se dostanou rovnou do místa v přepínači, kde jsou všechny analyzovány a zpracovány. Tím se vyhne celému procesu kontroly.



Obr. 3.2: Zabezpečený kanál [26]

Druhý způsob připojení je realizován prostřednictvím portu K, který je součástí datové roviny. Příchozí zpráva projde kontrolou (Packet matching function) a dojde k vytvoření tabulek toků. Následně je zpráva odeslána skrze virtuální port LOCAL do zabezpečeného kanálu, kde je dále zpracována. [26]

3.3 OpenFlow zprávy

OpenFlow ke komunikaci využívá tři typy zpráv: controller-to-swittch , asynchronní a symetrické.

3.3.1 Controller-to-Switch

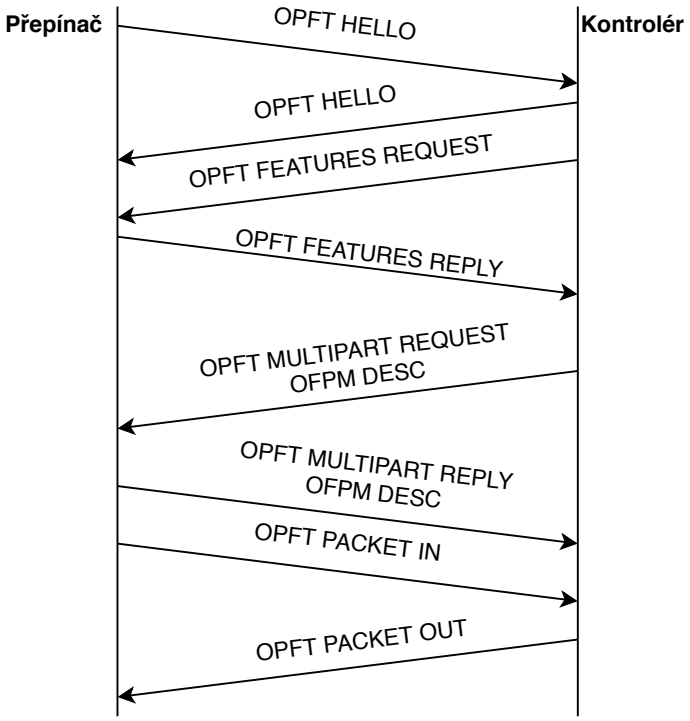
Jedná se sbírku zpráv, které umožňují kontroléru správu přepínače, zejména jeho stavů. Mezi tyto zprávy patří:

- **Features:** Kontrolér se dotazuje zprávou features request na identitu a základní vlastnosti přepínače. Ten zpětně odešle zprávu features reply, ve které se budou nacházet vyžádané informace. Této zprávy se využívá při vytváření kanálu OpenFlow.
- **Configuration:** Jedná se o zprávu, kterou kontrolér buď získává konfigurační parametry přepínače, anebo nastavuje jeho konfiguraci.
- **Modify-State:** Hlavní úlohou těchto zpráv je správa OpenFlow tabulek, například vkládání, mazání jednotlivých záznamů nebo nastavení portů. Zpráva je odesílána, když kontrolér potřebuje udělat změnu v tabulce. Tato změna říká přepínači co má dělat v případě, že v budoucnu přijde stejný paket. Skládá se z položek: Buffer ID (odkaz na paket), Idle Timeout (Čas po který ukládá datový tok v mezipaměti. V případě, že nepřijde stejný paket se stejnou žádostí do daného časového intervalu záznam bude odstraněn), Hard Timeout (V případě vypršení tohoto času je záznam odstraněn), Action (Operace kterou/které má vykonat s daným paketem.), Priority (Udává prioritu paketu v tabulce toků.).
- **Read-State:** Slouží ke sběru informací z přepínače. Například konfigurace, statistiky nebo funkce prostřednictvím tzv. multipart message.
- **Packet-out:** Tyto zprávy se skládají z paketu nebo z buffer ID, které na něj odkazuje a ze seřazeného seznamu akcí. Tento seznam nesmí být prázdný jinak dojde k zahození paketu. Jedná se o zprávu, jenž obsahuje instrukce ke zpracování specifického paketu. Zprávy tohoto typu se využívají v případě, že je třeba odeslat zpráva přímo na specifický port přepínače, nebo slouží jako odpověď na zprávu Packet-in.
- **Asynchronous-Configuration:** Využívá se v případě odesílání asynchronní zprávy, a to k nastavení filtrování těchto zpráv. [27] [25]

3.3.2 Asynchronní zprávy

Jedná se o zprávy, které odesílají přepínače k informování o příchodu paketu, nebo o změně stavu přepínače. Níže jsou vypsány jednotlivé typy těchto zpráv:

- **Packet-in:** Za pomoci této zprávy se předává řízení paketů kontroléru, a to v případě, že nastane výjimka. Ta vznikne například v situaci, kdy se v tabulce toků nenachází záznam o daném paketu a přepínač požaduje po kontroléru informace o tom, jak s daným paketem zacházet. Tato zpráva se dále využívá když má příchozí paket v hlavičce údaj o tom, že v případě nenalezení shody v tabulce má být paket odeslán kontroléru. Na tuto zprávu reaguje zprávou Packet-out.



Obr. 3.3: Komunikace přepínač kontrolér [28]

- **Flow-Remowed:** Tato zpráva slouží jako potvrzení kontroléru o smazání záznamu z tabulky, anebo jako informování o vypršení časového limitu záznamu.
- **Port-status:** Slouží jako informační zpráva v případě, že dojde ke změně stavu portu přepínače. Je odeslána například tehdy, je-li port v základu nastaven jako aktivní a dojde k jeho manuálnímu vypnutí nebo k výpadku linky.
- **Role-status:** Odesílá se, když je v síti více kontrolérů a jeden z nich změní svou roli na master.
- **Controller-Status:** Slouží jako oznámení o změně stavu přenosového kanálu. Přepínač tuto zprávu odesílá všem kontrolérům. [6] [27]

3.3.3 Symetrické zprávy

Tyto zprávy jsou odesílány bez předchozího upozornění. Níže je soupis těchto zpráv:

- **Hello:** Tuto zprávu si vyměňuje kontrolér s přepínačem při zahájení spojení.
- **Echo:** Využívá se k ověření komunikace přepínače s kontrolérem a v případě, že je zařízení funkční, odešle tuto zprávu nazpět. Lze také využít při měření latence nebo šířky pásma.
- **Error:** Při zaznamenání chyby při přenosu odešle přepínač tuto zprávu kontroléru, aby ho obeznámil s problémem.

3.4 Zpracovávání zpráv

Odesílání zpráv v OpenFlow je spolehlivé, ale automaticky nedochází k potvrzování zpracování zpráv. Je realizováno buď hlavním, nebo pomocným připojením. K doručení zprávy dochází téměř vždy až na případ, kdy dojde k selhání přenosového kanálu. Všechny zprávy přijaté od kontroléru jsou zpracovány přepínačem. Pokud ji nelze zpracovat, je odesláno chybové hlášení. U zpráv packet-out nemusí dojít ke zpracování, což může vést k jeho zahození z důvodu zahlcení aplikací pravidla QoS, nebo pokud je odeslán na špatný, či blokovaný port. Další povinností OpenFlow přepínače je odesílání asynchronních zpráv o změně stavu protokolu (odstranění toku, stav portu, atd.), které mohou být filtrovány (například nastavením podmínek, asynchronní konfigurací).

Zprávy se mohou shlukovat do tzv. Bundle messages. Bundle message je skupina zpráv, která se chová jako jedna. Zprávy obsažené ve skupině jsou vyhodnocovány jako celek. V případě, že je jedna ze zpráv chybná, je chybná i celá skupina.

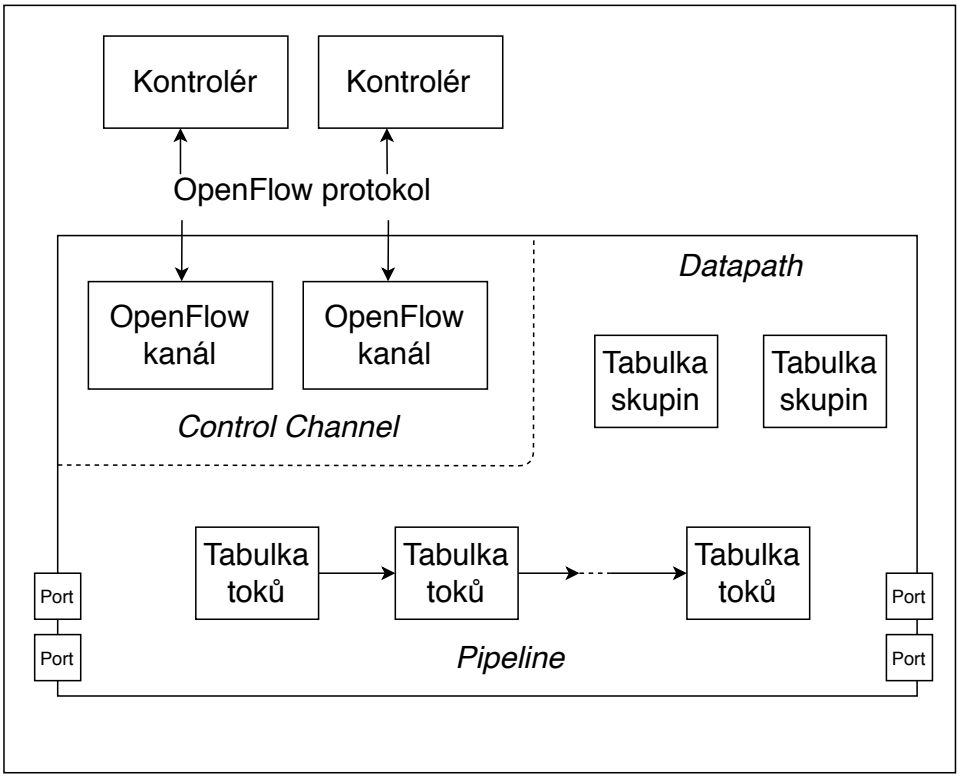
Uspořádání zpráv může být realizováno buď za pomoci tzv. barrier messages, nebo pomocí přepínače tak, aby výkon byl co nejvyšší. Barrier message je zpráva, kterou odesílá kontrolér když chce zasáhnout do procesu zpracovávání a uspořádání zpráv v přepínači. Další funkcí je poskytnutí potvrzení o dokončení operací příslušných zpráv. Využívají se nejčastěji v případě, kdy dvě po sobě jdoucí zprávy jsou vzájemně závislé a musí být mezi ně vložena barrier message. [27]

3.5 OpenFlow kontrolér

OpenFlow kontrolér je entita, která za pomoci softwaru řídí OpenFlow přepínače. Mezi jeho základní funkce patří poskytování náhledu sítě a řízení síťového provozu. Dále umožňuje vkládání, mazání a úpravu datových toků v tabulkách přepínače. Kontrolér je k přepínači připojen přes managment port, který se nachází v instanci VRF (Virtual Rounting and Forwarding) a poskytuje kontroléru zabezpečené připojení. Ve většině případů běží na serveru Linux. [29]

3.6 OpenFlow přepínač

Vnitřní struktura OpenFlow přepínače je zobrazena na obr. 3.4 Skládá se z jedné, nebo více tabulek toků, tabulky skupin a tabulky ve které jsou uloženy záznamy měření. Účelem těchto tabulek je provádět operace s pakety, například směrování, nebo vyhledávání. Dále se zde nachází jeden, nebo více kanálů, které jsou využity pro komunikaci mezi kontrolérem a přepínačem. Komunikace probíhá za pomoci OpenFlow protokolu.



Obr. 3.4: OpenFlow Přepínač [27]

Logika OpenFlow přepínače tkví v tom, že příchozí paket je porovnán s tabulkou toků a v případě, že najde shodu, vykoná na základě priority příslušnou operaci. Paket, který úspěšně prošel kontrolou, může být poslán příslušným portem dál, nebo může být provedena kontrola, či případná úprava hlavičky. Poté lze pokračovat, nebo může dojít k zahození paketu. V případě, že paket kontrolou neprojde (v tabulce není nalezena žádná shoda) závisí rozhodnutí na konfiguraci. Může být podroben kontrolou skrze další tabulky, zahozen, nebo odeslán ke kontroléru za pomoci zabezpečeného kanálu. Pravidla jsou stanovena kontrolérem. [24] [27]

3.6.1 Porty

Jedná se o logicky propojené rozhraní, které umožňují spolu s OpenFlow komunikaci mezi jednotlivými přepínači a zbytkem sítě. Přepínače je rozdělují na tři skupiny: fyzické, logické a rezervované porty.

Fyzické porty

Jedná se o porty, které jsou definované hardwarovým rozhraním přepínače. V případě, že je přepínač virtualizovaný, mohou tyto porty značit virtuální řez daným

rozhraním přepínače.

Logické porty

Tyto porty neodpovídají přímo jednotlivým fyzickým rozhraním přepínače. Jsou na vyšší úrovni abstrakce a mohou být definované uvnitř přepínače bez užití metod pro OpenFlow. Mezi takové metody patří například linková agregace, tunelování, nebo loopback rozhraní. Umožňují paketizaci dat a mapování na fyzické porty. Operace prováděné logickými porty závisí na softwarové implementaci. Zároveň musí být viditelné operacím OpenFlow a musí s nimi komunikovat jako s fyzickými porty.

U těchto portů je komunikace realizována za pomoci softwarové komponenty zvané Tunnel-ID. Příchozí paket je odeslán kontroléru zároveň s údaji o daném fyzickém a logickém portu.

Rezervované porty

Rezervované porty určují obecné úkony OpenFlow přepínače, například odesílání zpráv kontroléru, zaplavování, přeposílání za pomoci metod, které nevyužívají OpenFlow. Existuje mnoho druhů s tím, že přepínač musí mít podporu pro porty s tímto označením: ALL, CONTROLLER, TABLE, IN_PORT, ANY. Další, které už nejsou vyžadovány jsou například: LOCAL, NORMAL, FLOOD. [27]

3.6.2 Tabulky

Tabulka toků

Přepínač obsahuje tabulku jednotlivých datových toků. Ty využívá k porovnání a zpracování jednotlivých paketů. Mají přidělené priority, podle kterých se rozhodne co se s daným paketem stane. Datový tok je při překročení časového limitu odstraněn.

Když přepínač obsahuje více tabulek, zřetězí se do tzv. pipeline. Pipeline je soustava propojených tabulek toků, která poskytuje párování, přeposílání a další úpravy paketů. Tabulky jsou číslovány od nuly. V případě příchodu paketu na vstupní port dojde nejprve ke kontrole položek nulté tabulky a na základě výsledku i ke kontrole položek dalších tabulek. Podle výsledku kontroly může dojít ke zpracování dalšími tabulkami s vyšší hodnotou za pomoci akcí, přeposlání na výstupní port, či k zahození paketu. Každý tok v tabulce obsahuje následující informace: pole shody (pole se kterým je porovnáván příchozí paket), priorita (udává pořadí toků, čím vyšší je to hodnota tím vyšší má prioritu), čítače, instrukce (operace, která přeposílá paket na port, do následující tabulky, nebo upravuje pole paketu), časový limit, cookie a příznaky (flags). [27] [29]

Pole shody	Priorita	Čítače	Instrukce	Časový limit	Cookie	Příznaky (flags)

Obr. 3.5: Informace datového toku [27]

Tabulka skupin

Tato tabulka obsahuje skupiny vstupů. Nachází se zde skupiny vstupů, kterým je přidělen typ směrování, a to buď všesměrové, nebo vícesměrové. Každý vstup obsahuje následující informace: identifikátor skupiny, typ skupiny, čítače, balíčky akcí (Action Buckets). Každá skupina musí obsahovat minimálně 1 balíček akcí. V případě, že balíček nemá, dojde k zahození paketu. Vstupy skupin lze rozdělit na 4 typy:

- All: vykoná všechny skupiny akcí v tabulce skupin
- Select: vykoná jednu skupinu akcí v tabulce skupin
- Indirect: vykoná jednu definovanou skupinu akcí v tabulce skupin
- Fast failover: vykoná první spustitelnou (s aktivním portem) skupinu akcí v tabulce skupin

[27]

Tabulka měření

Z procházejícího datového provozu lze snímat parametry, například rychlost, nebo šířka pásma. Díky tabulce měření je umožněno tyto parametry ovlivňovat, a to implementováním omezovačů rychlosti, nebo za použití QoS. Každý vstup tabulky obsahuje následující informace: identifikátor měření, pásmo měření a čítače. Pásma měření specifikují rychlost a chování (pokles, nebo změna DSCP). Jakmile dojde ke shodě paketu se vstupem měření, měřící pásmo s nejvyšší nakonfigurovanou rychlostí, která je vyšší jak současná měřená, se použije. [27] [31]

3.7 Verze OpenFlow

OpenFlow protokol se v průběhu času vyvíjel od verze 1.0, kde se nacházela pouze jedna tabulka toků a 12 polí s pevnou shodou, až po poslední verzi, která nabízela tabulky toků, 42 polí s pevnou shodou a další funkce. Podrobnější specifikace je zmíněna níže. V této sekci je citováno z [30]

3.7.1 Verze 1.0

Tato verze vyšla v prosinci roku 2009. Obsahuje pouze jednu tabulku toků, jejíž součástí jsou vstupy toků, které obsahovaly tři komponenty: Pole hlavičky, čítače

a akce. Pole hlavičky obsahovalo dvanáct pevných porovnávacích prvků. Z důvodu omezení na jednu tabulku toků byly OpenFlow přepínače schopny vykonávat pouze jednu operaci. Pro implementaci QoS (Quality of Services) poskytuje volitelnou akci „enqueue“.

3.7.2 Verze 1.1

Oproti verzi 1.0 zde přibilo více tabulek toků a tzv. tabulky skupin, které zvýšily počet operací, jež mohl přepínač vykonat. Další změnou bylo přejmenování pole hlavičky a akce, na pole shody a instrukce. Přidáním více tabulek bylo zpracování rozdělené do dvou kroků: učení MAC adres a VRF (Virtual Routing and Forwarding).

3.7.3 Verze 1.2

S touto verzí přišla struktura TLV (Type-Lenght-Value), která umožňuje vkládání polí shody modulárnějším způsobem. Těmto polím se říká OXM (OpenFlow Extensible Match). Díky OXM jsou kritéria shody přijímány rychleji, čímž je zvýšena flexibilita polí shody. Dále je přidána podpora IPv6 založená na OXM.

3.7.4 Verze 1.3

Tato verze přináší tabulku s názvem Meter table (tabulka měření více viz. 3.6.2), která rozšiřuje možnosti QoS. Dále rozšířila tabulku toků o tabulku špatných vstupů (table-miss entry). V předcházejících verzích byl paket zahozen, nebo odeslán kontroléru formou packet-in zprávy. S touto tabulkou je zpracovávání chování paketu bez shody flexibilnější. Tato tabulka definuje skupinu akcí, které jsou provedeny v případě, že paket nenajde shodu.

3.7.5 Verze 1.4

Ve verzi 1.4, kromě předchozích inovací, přibyla Synchronizační tabulka. S tou mohou být tabulky toků synchronizovány dvěma způsoby: obousměrně nebo jednosměrně. Pokud se jedná o obousměrnou synchronizaci, změny provedené kontrolérem se musí odrazit ve zdrojové tabulce. Další změnou je přidání funkce „Bundle“. Tato funkce umožňuje seskupit stavové modifikace do tzv. transakční skupiny. V případě použití bundle jsou ve skupině buď použity všechny modifikace, nebo žádná. Lze toho využít při odesílání většího množství zpráv na více přepínačů.

3.7.6 Verze 1.5

V poslední verzi přibyl tzv Scheduled bundle (plánovaný balíček), který oproti klasickému bundle obsahuje položku čas provedení. Přepínač, který obdrží tento bundle, použije zprávy v čase nejbližšímu času provedení. V poslední řadě byli přidány výstupní tabulky (Egress table). Na základě této tabulky je umožněno odeslat paket správným výstupním portem.

4 Open source kontroléry

V této sekci je vytvořený výčet dostupných open source kontrolérů a popsány jejich základní vlastnosti. Následně v praktické části je vybrán jeden z nich, který je popsán podrobněji.

4.1 Ryu

Tento kontrolér byl navržen tak, aby zvýšil efektivitu sítě a usnadnil její správu. Disponuje dobře definovanými rozhraními API, čímž je usnadněna správa a řízení sítě. Díky tomu je možné ho jednoduše přizpůsobit požadavkům jednotlivých aplikací. Využívá se nejčastěji v malých podnicích a pro výzkumné aplikace.

Zdrojový kód Ryu je psaný v programovacím jazyce Python. Ryu je umístěný spolu s jeho zdrojovým kódem na GitHubu Je možné ho upravovat a bezplatně využívat. Neposkytuje vysokou modularitu, je centralizovaný a nelze ho provozovat napříč platformami. Je podporovaný pouze Linuxem, čímž je omezeno jeho použití v reálných tržních aplikacích. Ke konfiguraci využívá webové rozhraní. Jižní rozhraní podporuje protokoly NETCONF, OF-config a OpenFlow (verze 1.0 - 1.5). Severní rozhraní podporuje pouze REST API pro jižní rozhraní. [12] [13]

4.2 ONOS (Open Network Operating System)

Tento kontrolér je napsaný v programovacím jazyce Java. Jižní rozhraní tohoto kontroléru podporuje protokol OpenFlow ve verzích 1.0 a 1.3 a řídící protokoly jako je NETCONF a PCEP. Disponuje webovým rozhraním a poskytuje vysokou modularitu. Nejčastěji se využívá v data centrech a ve WAN. Mezi jeho hlavní výhody patří výkon a skutečnost, že se jedná o distribuovaný kontrolér. [12] [17]

4.3 NOX

Jedná se OpenFlow kontrolér, který slouží jako platforma pro řízení, správu sítě a vývoj síťových řídících aplikací. Byl vyvinut společností Nicira v roce 2009. Je navržen ve třech základních verzích: NOX, NOX classic a POX.

NOX je vytvořený v programovacím jazyku C++, je rychlejší a má lepší kódovou základnu. Poskytuje nízkou modularitu, je centralizovaný a podporovaný nejčastěji systémem Linux. Jeho grafické rozhraní (GUI) je programované v Pythonu. Jeho jižní rozhraní podporuje pouze OpenFlow verze 1.0. Nejčastěji se používá v akademickém síťovém výzkumu a vývoji aplikací SDN. [12] [26]

4.4 POX

POX je realizovaný společně s jeho grafickým rozhraním za pomoci jazyka Python. Poskytuje nízkou modularitu, je centralizovaný a může být realizovaný například na Operačním systému Linux, MAC a Windows. Jeho jižní rozhraní podporuje OpenFlow verze 1.0. Nejčastěji se využívá v univerzitních sítích. [12]

4.5 Trema

Trema je framework, který se využívá k vývoji OpenFlow kontrolérů. Na rozdíl od ostatních kontrolérů tento model poskytuje základní infrastrukturní služby jako součást základních modulů. Uživatel má svobodu v konfigurování OpenFlow kontroléru. Moduly lze vytvářet za pomoci programovacího jazyka Ruby nebo C. Poskytuje jednoduchý OpenFlow ovladač sloužící k řízení OpenFlow zpráv. Podporuje OpenFlow verze 1.3. Využívá se nejčastěji v kampusových sítích [12] [6]

4.6 Floodlight

Floodlight je centralizovaný kontrolér, který vytvořila společnost Big Switch Networks. Je naprogramován v programovacím jazyce Java. Má modulární architekturu, která poskytuje správu sítě a zařízení, výpočet trasy apod. Podporuje ho většina operačních systémů. Jeho jižní rozhraní podporuje OpenFlow verze 1.0 a 1.3. Nejčastěji je využíván v univerzitních sítích. [12]

4.7 OpenDaylight

Jedná se kontrolér využívaný primárně v data centrech. Jeho poslední aktualizace však poskytuje jiné využití. Je realizovaný programovacím jazykem Java. Jižní rozhraní podporuje spoustu protokolů, například OpenFlow ve verzích 1.0, 1.3, 1.4, NETCONF/YANG, OVSDB, PCEP, BGP-LS, LISP, SNMP, atd. Tento kontrolér může být realizován na operačních systémech Linux, MAC a Windows. Je distribuovaný a poskytuje vysokou modularitu. Díky implementaci nových modelů se stal jedním z kontrolérů, který podporuje IoT. [12]

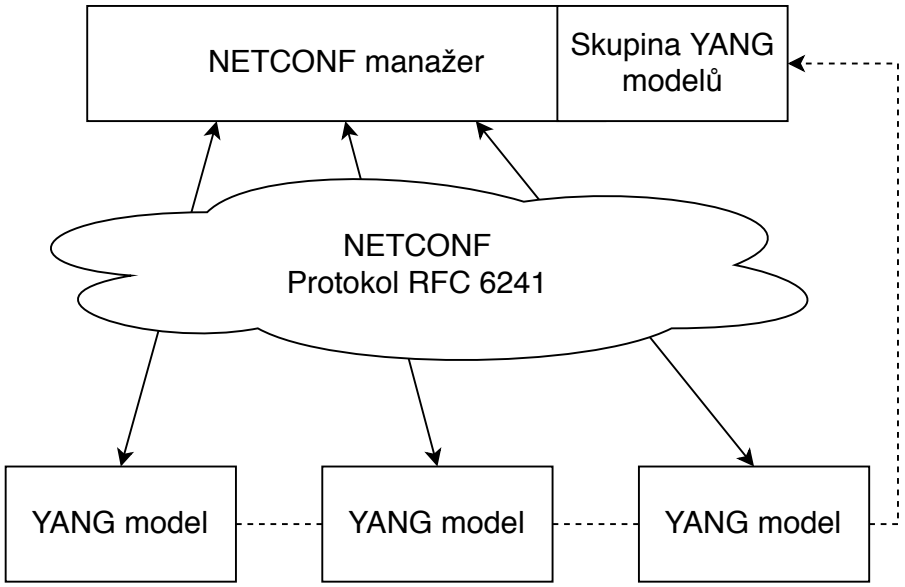
4.8 Beacon

Kontrolér který byl vytvořen v programovacím jazyce Java a obsahuje webové rozhraní. Poskytuje modulární architekturu, která na svém jižním rozhraní podporuje

OpenFlow verze 1.0. Podporuje ho většina operačních systémů. Nejčastěji se využívá k výzkumným účelům. [12]

5 NETCONF

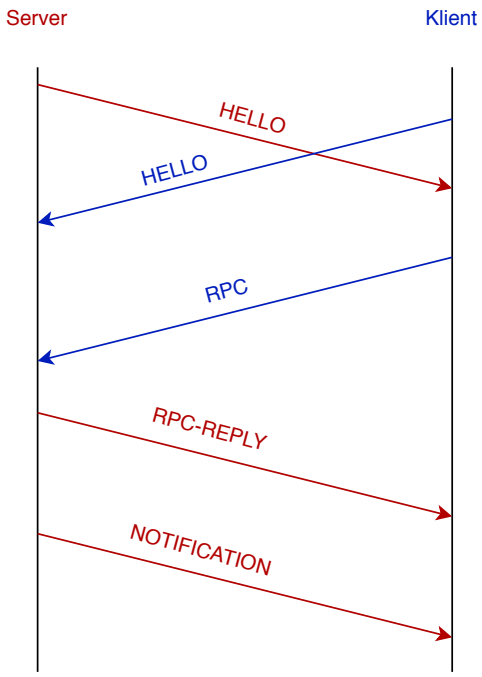
Jedná se o spojově orientovaný protokol, který se využívá ke správě konfigurace síťových zařízení. Byl vyvinut v roce 2006 společností IETF. Nahradil dříve používaný protokol SNMP, který umožňoval pouze čtení stavů sítě. Hlavním důvodem vzniku NETCONF a YANG bylo vytvořit systém správy sítě, spravující síť na servisní úrovni, která zahrnuje: Standartizované datové modely (YANG), konfigurační transakce v celé síti, ověřování funkčnosti konfigurace, centralizovanou zálohu a obnovu konfigurace. Software, který tuto funkčnost zajišťuje se nazývá NETCONF manager, jehož funkčnost je znázorněna na obrázku 5.1. [6] [32] [33]



Obr. 5.1: Síťový diagram systému řízení NETCONF [33]

Komunikace NETCONF probíhá na bázi klient-server, kde klient je skript nebo aplikace, která běží jako součást síťového manažera a server je síťové zařízení (router, switch). K vykonávání poskytovaných operací využívá princip RPC (Remote Procedure Call), který umožňuje vzdálené aplikování instrukcí na jiném místě v síti. Odesílaná data jsou šifrována za pomoci jazyku XML (Extensible Markup Language). Samotná komunikace probíhá následujícím způsobem: [6] [32] [33]

Komunikace se zahajuje odesláním zprávy hello, kdy se klient a server dohodnou na verzi protokolu a server odešle poskytované operace. V další části klient odešle zprávu rpc, jenž obsahuje specifikace a identifikátor dané operace, kterou che využít. V případě, že nedojde k potížím, server odešle odpověď rpc-reply, kde se daná operace již nachází. V případě chyby odešle server zprávu rpc-error. Dále může server odesílat informace o specifických událostech, které nastaly. Těmto zprávám potom



Obr. 5.2: Komunikace NETCONF [34]

říkáme notifikace. [34]

5.1 Poskytované funkce

NETCONF a YANG řeší nedostatky SNMP a přidávají k již zmíněnému čtení stavů sítě další funkce, například:

5.1.1 Konfigurační transakce

Konfigurace je založená na atomových transakcích, což je skupina nezměnitelných konfiguračních příkazů požadovaných ke změně sítě ze stavu A do B. K úspěšnému provedení transakce je nutné, aby byly úspěšné všechny příkazy nezávisle na pořadí jejich aplikování. Tudíž neexistuje jiný stav než A (selhání některého z příkazů), nebo B (úspěch transakce jako celku). [33]

5.1.2 Souběžná aktivace konfigurace v celé síti

NETCONF umožňuje nastavení a zároveň aktivaci konfigurace na více síťových zařízeních najednou. Například pokud budeme chtít nakonfigurovat VPN na všech zařízeních najednou, potom NETCONF umožní distribuci, ověření, uzamčení, potvrzení

a aktivaci konfigurace na zařízení. Výsledkem této sady akcí bude synchronizované nastavení VPN. [33]

5.1.3 Ověření funkčnosti konfigurace

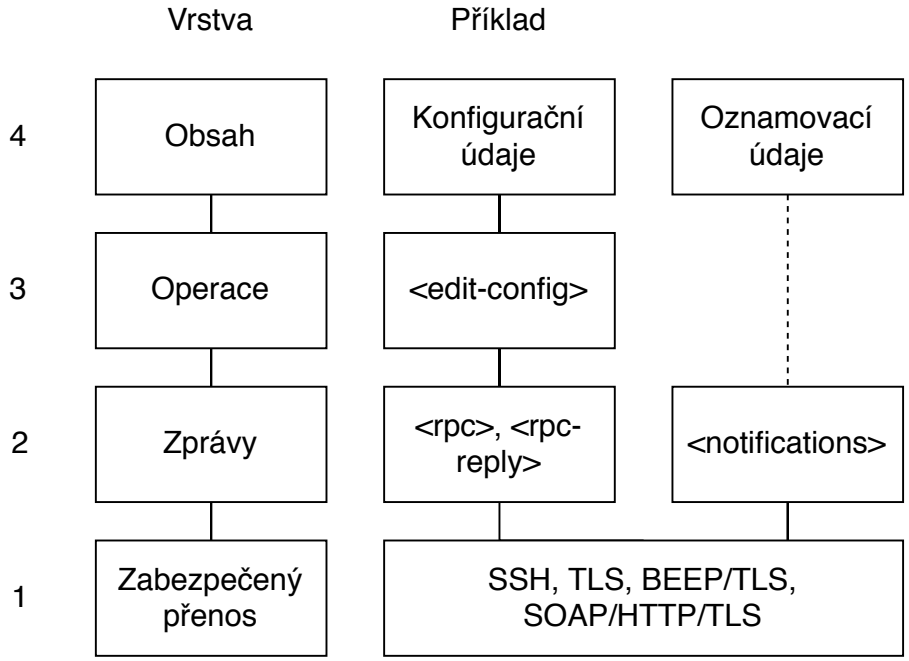
NETCONF server si ukládá tzv. „databázi uchazečů“, pomocí níž lze implementovat konfigurační transakci v celé síti a zároveň ověřit její funkčnost. K ověření funkčnosti dojde po odeslání a aktivaci konfigurace u všech uchazečů. V případě neuspokojivých výsledků lze všechny změny vrátit zpět. [33]

5.1.4 Ukládání a obnova konfigurace

NETCONF manager si ukládá zálohu konfigurace všech zařízení proto, aby ji v případně nutnosti bylo možné obnovit do původního stavu. [33]

5.2 Architektura

Architektura tohoto protokolu lze rozdělit na 4 vrstvy, které se nachází na 5.3.



Obr. 5.3: Architektura protokolu NETCONF [32]

- Ve spodní části se nachází zabezpečená transportní vrstva (Secure Transport), která poskytuje spojovanou komunikaci (TCP) mezi síťovým zařízením (server) a konfiguračními aplikacemi (klient). K zabezpečení komunikace a zajištění důvěryhodnosti a integrity dat používá většinou protokoly transportní vrstvy, například SSH nebo TLS. [32]
- Vrstva zpráv (Messages) poskytuje jednoduchý rámcový mechanismus pro šifrování RPC zpráv a oznámení. [32]
- Vrstva operací (Operations) definuje sadu základních procesů serveru, například: get, get-config, edit-config, copy-config, atd. Jednotlivé operace jsou napsány ve formátu XML. [32]
- Vrstva obsahu (Content) obsahuje konfigurační a stavová data. Konfigurační data obsahují záznam o konfiguraci daného zařízení. Rozdělují se do třech odvětví: running (aktuální konfigurace), candidate (Zde se nachází konfigurace, která je připravena k nahrání) a startup (konfigurace, která se nastaví při startu zařízení). Stavová data obsahují informace o stavu komunikace (počet přenesených paketů, kapacitu sítě, atd.) Tato vrstva je popsaná pomocí YANG modelů. [32]

5.3 Rozšíření (Capabilities)

Jedná se o sadu funkcí, která doplňuje základní operace protokolu NETCONF. Server při navazovaní spojení říká jakými rozšířeními disponuje. Každé rozšíření je označené jednoznačným URI (Uniform Resource Identifier). Mezi základní rozšíření patří: Writable-Running (slouží k přepisování konfigurace systému), Candidate Configuration, Confirmed Commit, Rollback-on-Error (v případě podpory tohoto rozšíření umožňuje obnovu dat při chybné konfiguraci), Validate, Distinct Startup (umožňuje oddělení datových úložišť), URL, XPath. [32]

5.4 YANG

Jedná se o jazyk, používaný k modelování dat pro protokol NETCONF. Má hierarchické rozdělení podobné stromu, kde každý uzel obsahuje jméno, hodnotu a sadu dalších uzlů. Lze ho aplikovat na operace založené na NETCONF včetně konfigurace, stavů dat, RPC a oznámení. Díky tomu je zajištěn popis všech dat vyměňovaných mezi klientem a serverem. [35]

Rozděluje se na nadřazené moduly a podřízené moduly (obsahují materiál pro nadřazené). Každý z nich je rozdělen na tři části:

- Záhlaví: Obsahuje informace o modulu

- Revizní část: Udává informace o historii modulu
- Definiční část: Zde je definován datový model

Více informací o tomto modelu lze najít v dokumentaci viz. [35]

6 Praktická část

Na základě poznatků zjištěních v předchozí části byla realizovaná praktická část. Jedná se o výběr OpenFlow kontroléru a shrnutí jeho nejdůležitějších vlastností. V další části budou zmíněny topologie sítě laboratorní úlohy a jejich chování. Následně bude vytvořen seznam použitelné technologie. Dále pak bude popsána realizace těchto úloh a simulace. Na konci v příloze bude vytvořen laboratorní návod. V laboratorní úloze budou použity specifické nástroje, které se dosud v laboratorních cvičeních nevyužívali. Studenti budou potřebovat chvíli, aby se s těmito nástroji seznámili, proto doporučuji úlohu rozdělit na 2 bloky: část 1 seznámení se s prostředím a část 2 + 3 vlastní úloha.

6.1 Výběr kontroléru

V této části se zaměříme na open source kontrolér, který bude využit v laboratorní úloze. V teoretické části byly zmíněny dostupné kontroléry a jejich popis. V tabulce níže je připomenutí open source kontrolérů, ze kterých bylo vybíráno.

Tab. 6.1: Tabulka dostupných kontrolérů

Kontrolér	**Jazyk**	**Oblast využití**
Ryu	Python	Kampusové sítě, Výzkumné účely Učební pomůcka
ONOS	Java	Datacentra, WAN
NOX	C++	Kampusové sítě
POX	Python	Kampusové sítě
Trema	C	Kampusové sítě
Floodlight	Java	Kampusové sítě
OpenDaylight	Java	Datacentra
Beacon	Java	Výzkum

Z této škály kontrolérů byl vybrán kontrolér Ryu, který je svými vlastnostmi nejvhodnější pro realizaci této úlohy. Jeho vlastnosti budou popsány níže.

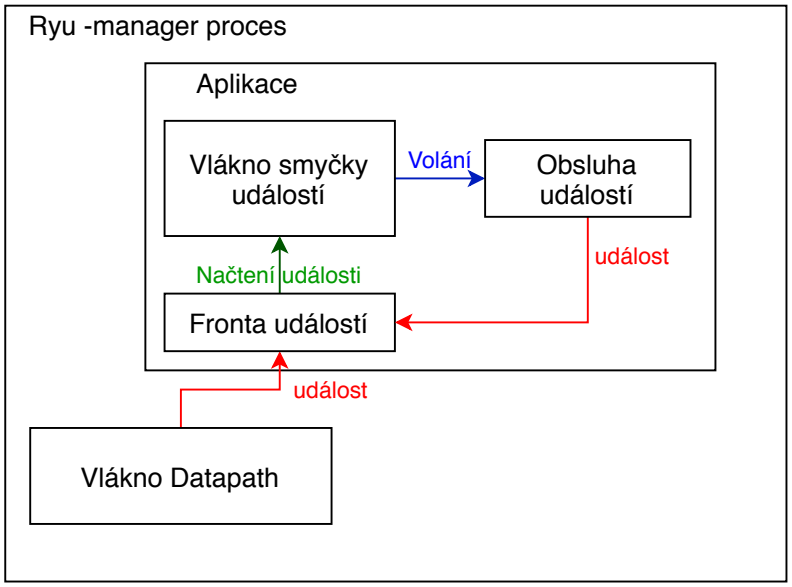
6.1.1 Ryu

Tento kontrolér se přesně hodí pro mou práci, protože obsahuje moduly, které lze využít k realizaci úlohy. Tyto moduly budou popsány níže. Dále disponuje funkcemi, které lze využít k experimentování, nebo ke školení. Zde je soupis jeho vlastností:

- Podpora OpenFlow 1.0 - 1.5
- Používá programovací jazyk Python, který je snadno naučitelný a lze ho využít ve spoustě případů
- Obsahuje framework založený na komponentách
- Jeho kód je volně dostupný pod licencí Apache 2.0.
- Poskytuje část kódu klíčových síťových protokolů OpenFlow
- Poskytuje platformu pro testování přepínačů s rozsáhlou sadou testů, pokrývající téměř každou funkci ve specifikacích OpenFlow.
- Ke konfiguraci zařízení využívá REST API.
- Disponuje obsáhlou dokumentací. [15] [16]

Architektura

Skládá se z balíku aplikací, které ke komunikaci využívají tzv. události. V ryu jsou popsány jako třída objektů, jenž dědí z „ryu.controller.event.EventBase“. Jednotlivé události se dočasně shromažďují ve frontách. Lze vytvořit tzv. Event handler, kterému lze přiřadit specifickou rutinu (činnost). Jakmile je spuštěna událost, odpovídající danému schématu, je zavolána příslušná rutina, která událost zpracuje. Načítání a spouštění aplikací je realizováno skrze spustitelný soubor ryu-manager. Ryu se skládá z více vláken a umožňuje měnit průběh instrukcí kódu za běhu, za pomocí tzv. eventless. Architektura kontroléru se nachází na 6.1. [14]



Obr. 6.1: Architektura Ryu [14]

Moduly Ryu

Mezi základní moduly Ryu patří: Traffic Monitor, REST Linkage, Link Aggregation, Spanning Tree, Firewall, Router, QoS, OpenFlow Switch Test Tool. Některé z těchto modulů jsou popsány níže. V této sekci je čerpáno ze zdroje [14].

Traffic Monitor

Od sítí se očekává plynulý a stabilní provoz, ale i tak se objevují problémy. Aby se tyto problémy včas identifikovali a řešili, je síť monitorována a probíhá získávání informací o jejich parametrech. K monitorování je využíván modul Traffic Monitor. Kontroluje záznamy jako jsou vstupy toků, hodnoty Table-miss, stavy jednotlivých portů, statické informace (čísla portů, počet odeslaných a přijatých paketů, počet zahození, atd.). Kontrolu provádí každých deset vteřin. Traffic Monitor je implementovaný ve třídě SimpleMonitor13. Aby bylo možné monitorovat síť a zároveň zpracovávat pakety, je nutné používat více vláken.

Firewall

Firewall udává souhrn pravidel pro síťový provoz. Pravidla lze pouze nastavit a smazat, ale ne upravovat. S přidáním se pravidlu automaticky přidělí ID. Dále je nutné nastavit prioritu podle, které bude pravidla odlišovat. Při příchodu paketu na daný vstup, kde jsou pravidla nastavena, jsou pravidla s vyšší prioritou vyhodnocena přednostně. Pravidla firewallu jsou odesílána na zařízení skrze REST API a k jejich definování je využita aplikace curl.

Router

Tento modul slouží k nastavení směrování provozu v síti. Pravidla lze nastavit a smazat, nikoliv upravovat. K přidělování pravidel se používá jako identifikátor switchID, který říká o jaký přepínač se jedná. Mezi hlavní patří nastavení IP adresy směrovače, default route a statické směrování. K nastavení bude využit webový server, který běží na portu 8080. Data odesílá pomocí REST API. K zadávání příkazů se používá taktéž linuxová aplikace curl a její příkaz POST. Ke smazání se používá příkaz DELETE. Za tyto příkazy je zapsán obsah ve formátu json, který vypadá následovně:

```
{"key":"value"} http://localhost:8080/router/switchID/vlan
```

SimpleSwitch

Jedná se o software, který ovládá přepínač. Jeho funkce je následující: Po registraci přepínače do kontroléru se do tabulky toků přepínače přidá první tok typu „catch-all“ s nízkou prioritou, který dává pokyn k odesílání všech paketů do kontroléru. Pro tento přepínač se vytvoří tabulka MAC-to-Port. Jakmile přijde na vstup přepínače paket, odešle se zpráva PacketIN kontroléru a do tabulky MAC-to-Port se přidá položka přiřazující zdrojovou MAC adresu přijímacímu portu. Kontrolér se podívá, jestli má cílovou adresu ve své tabulce. Pokud ne, donutí přepínač odeslat záplavu. Pokud ano, nastaví cílový port na port přidružený k cílové MAC adrese, přidá do přepínače tok (s prioritou vyšší než má předchozí), který se shoduje s aktuální MAC adresou a cílovým portem. Akce obsažená v tomto toku donutí přepínač odeslat paket portem, který se pojí s cílovou adresou, pokud by se znovu objevil na vstupu. Na konci odešle kontrolér zprávu PacketOUT, která obsahuje pokyn co má přepínač s paketem udělat (buď záplava nebo odeslání paketu na daný port). [38]

6.2 Nástroje použité při tvorbě laboratorní úlohy

V této kapitole budou popsány jednotlivé softwarové komponenty, které budou využívány v praktické části. Jmenovitě půjde o VMware, OpenVSwitch, Wireshark.

6.2.1 VMware

Pro tuto úlohu bude použit jako hypervizor nástroj VMware. Používá se na platformách Windows a LInux a slouží k realizaci virtualizace, nebo-li k vytváření a propojování více virtuálních strojů na jednom fyzickém zařízení. Mezi jeho hlavní funkce patří právě vytváření virtuálních sítí, sdílení souborů host-host, zálohování a klonování virtuálních strojů, čímž je usnadněna implementace více podobných strojů na různých místech. [39]

Konfigurace síťového připojení

- Bridge Networking - Slouží k propojení virtuálního stroje s hostujícím zařízením a sítí, ve které se nachází pomocí síťového adaptéru. Zjednodušeně propojuje virtuální stroj s fyzickým síťovým adaptérem. V tomto stavu je také propojen se všemi zařízeními, které mají stejný typ síťového připojení.
- Network Address Translation - Tento typ připojení umožňuje virtuálnímu stroji být v privátní síti, tudíž není dostupný pro ostatní sítě bez použití NATu. Využívá se v případě, že virtuální stroj potřebuje přístup k internetu.

- Host-Only Networking - Izoluje virtuální síť od ostatních. Hostující zařízení a virtuální stroj se nachází ve stejné síti. Propojení mezi nimi je realizováno virtuálním síťovým adaptérem, který je viditelný na hostujícím operačním systému.
- LAN Segments - Jedná se o privátní síť, která je sdílená ostatními virtuálními stroji. Využívá se při testování a síťové analýze. [39]

6.2.2 OpenVSwitch

Jedná se o vícevrstvý softwarový přepínač licencovaný licencí Apache 2. Podporuje správu rozhraní, směrovací funkce k programovému rozšíření a řízení. Hojně se využívá ve virtuálním prostředí. Poskytuje virtuální rozhraní, pomocí nichž lze propojovat virtuální stroje, další virtuální přepínače a servery. Může být provozovaný na řadě Linuxově založených virtualizačních nástrojích, například: Xen/XenServer, KVM a VirtualBox. [40]

OpenvSwitch je napsaný v programovacím jazyce C a poskytuje následující funkce:

- Standard 802.1Q VLAN model s trunk a přístupovými porty
- Spojení NIC s nebo bez LACP na upsttream přepínači
- NetFlow, sFlow(R) a zrcadlení pro zvýšení viditelnosti
- OpenFlow 1.0 a spoustu rozšíření
- Vysokovýkonné přeposílání s využitím Linuxového modulu jádra atd. [40]

6.2.3 Komponenty

Mezi hlavní komponenty a nástroje patří:

- ovs-vswitchd - démon implementující přepínač, spolu s doprovodným linuxovým module jádra pro přepínání podle datového toku
- ovsdb-server - jednoduchá databáze, ze které ovs-vswitchd získává konfiguraci
- ovs-dpctl - nástroj pro konfiguraci modulu jádra přepínače
- ovs-vsctl - nástroj pro dotazování a aktualizaci konfigurace ovs-vswitchd
- ovs-appctl - nástroj, který slouží k odesílání příkazů spuštěným Open vSwitch démonům
- ovs-ofctl - nástroj pro dotazování a řízení OpenFlow přepínačů a kontrolérů
- ovs-pki - nástroj sloužící k vytváření a správě veřejného klíče infrastruktury pro OpenFlow přepínače
- tcpdump - umožňuje analýzu OpenFlow zpráv [40]

6.2.4 Wireshark

Jedná se o nejrozšířenější volně dostupný analyzátor síťových paketů. Podporuje analýzu široké škály síťových protokolů, od protokolu ICMP po protokol OpenFlow. Dokáže zachytit provoz velkého množství síťových rozhraní od Ethernetu po USB. Pomocí něj lze zjistit údaje v hlavičkách jednotlivých paketů, například zdrojovou a cílovou IP adresu apod. Další možnost využití je pro studijní účely, zejména ke sledování funkce jednotlivých protokolů (zachycení dílčích paketů, které jsou odeslány během komunikace). Wireshark je multiplatformní, lze ho používat jak na Linuxu, tak na Windowsu, či MACu. [41]

6.3 Topologie sítě

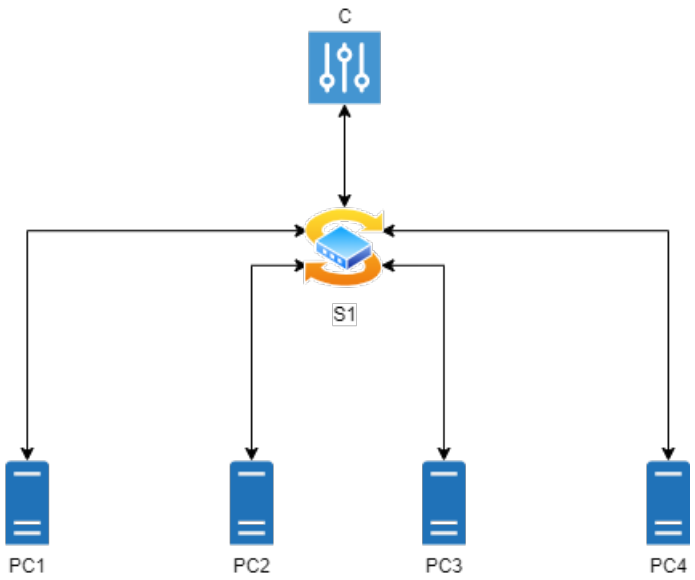
Tato sekce se bude zaobírat popisem jednotlivých topologií, které budou využity v laboratorní úloze. Jedná se o topologii s jedním přepínačem a čtyřmi hosty, dále pak o topologii využitou pro realizaci firewallu a topologii, která bude použita ke směrování pomocí modulu Router. K realizaci topologií budou vytvořeny dva virtuální stroje s OS Ubuntu, kde na jednom z nich bude umístěn kontrolér Ryu a na druhém bude Mininet spolu s OpenVSwitchem. Tyto dva virtuální stroje budou umístěny na serveru a následně propojeny pomocí virtuálního přepínače uvnitř serveru. Přistupovat se k nim bude za pomoci nástroje VMRC a to ze dvou PC v laboratorní učebně.

6.3.1 Přepínač se 4 hosty

Na obrázku 6.2 je zobrazena první ze síťových topologií, která bude využita k sestrojení laboratorní úlohy. Ve vrchní části se nachází kontrolér, jenž bude ovládat přepínač nacházející se pod ním. počítače PC1 až PC4 jsou připojeny k přepínači na portu 1 - 4 a budou sloužit k demonstraci funkčnosti softwarově definované sítě.

Pro realizaci topologie bude využit nástroj Mininet. Jedná se o nástroj, který umožňuje navrhnout a otestovat libovolnou topologii. K realizaci byl napsán zdrojový kód, který po jeho spuštění vytvoří topologii, jenž bude složena z jednoho přepínače s1 a 4 hostů h1 - h4. Každému z hostů bude automaticky po spuštění skriptu nastavena IP adresa (192.168.1.1 - 4/24) a MAC adresa (00:00:00:00:10:11 - 14). Dále bude přiřazen vzdálený kontrolér na IP adrese 10.100.107.10. Zdrojový kód je zobrazený v příloze A.3.

Topologie bude sloužit k demonstraci komunikace mezi kontrolérem, OpenFlow přepínačem a jeho hosty. Studenti by si na této topologii měli osvojit základní práci s kontrolérem, OpenVswitchem a Mininetem. Dále by měli zjistit, jaké zprávy mezi sebou kontrolér a přepínač odesílají, a to pomocí aplikace Wireshark.

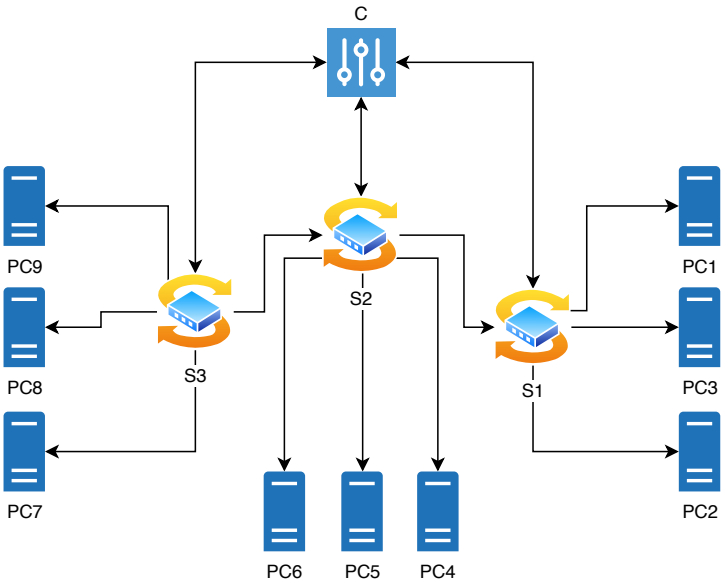


Obr. 6.2: Topologie sítě laboratorní úlohy

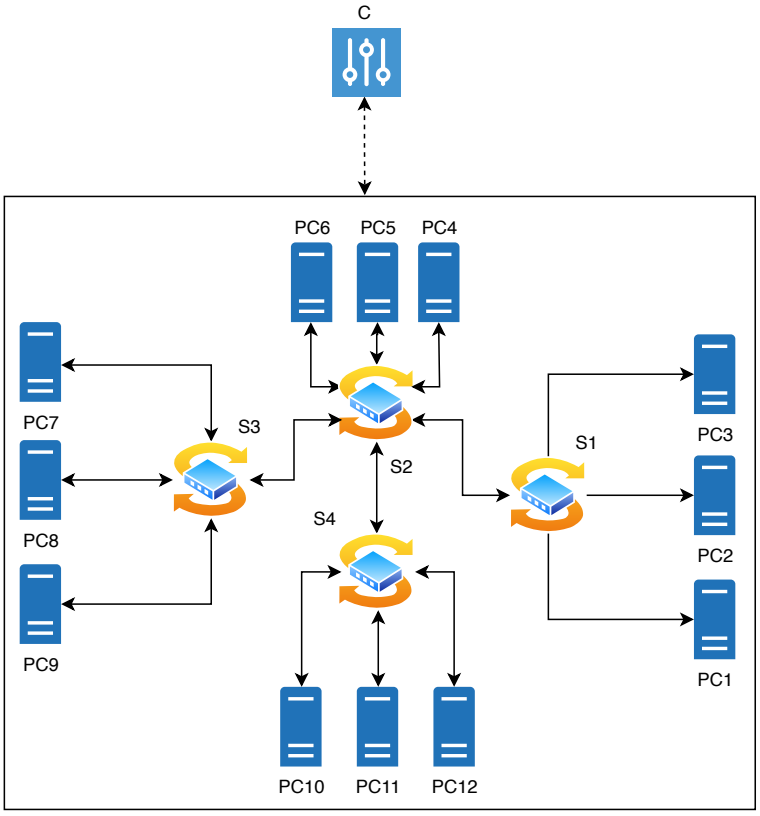
6.3.2 Topologie pro firewall

Na obrázku 6.3 je znázorněna topologie, která bude využita pro konfiguraci firewallu. Skládá se ze 3 přepínačů, kde každý z nich ovládá kontrolér a disponuje 3 stanicemi PC. všechny přepínače jsou spolu propojeny tak, že tvoří stromovou topologii.

V příloze A.1 je zobrazen zdrojový kód této topologie, který byl vytvořen pro Mininet. Přepínače byli vytvořeny pomocí příkazu **self.addSwitch**, jehož parametrem je název přepínače a stanice PC byli vytvořeny pomocí příkazu **self.addHost**, kde každá z nich má nadefinovanou MAC adresu **00:00:00:00:0m:0n**, přičemž **m** je číslo přepínače a **n** je pořadí PC. Každý z PC má nastavenou IP adresu **192.168.1.n**, kterou lze kdykoliv změnit. Propojení jednotlivých přepínačů mezi sebou a připojení hostů je realizováno pomocí příkazu **self.addLink**, kde uvnitř závorky je definováno co s čím je propojeno a na jakém portu. V případě, že není port zadán vygeneruje si ho automaticky. V hlavní větvi kódu je nastaven vzdálený kontrolér c1 na IP adrese **10.100.107.10** s portem **:6653**. Ve stejné větvi je nastaveno i spouštění celé topologie. Nejprve je přidělena třída, ze které má čerpat a kontrolér. Následně je příkazem **net.start** spuštěna. Veškeré toto nastavení je automaticky promítnuto do OVS. V poslední řadě je zde nastavena podmínka, která při vypnutí topologie vymaže celé nastavení z OVS.



Obr. 6.3: Topologie pro nastavení firewallu



Obr. 6.4: Topologie pro nastavení firewallu

6.3.3 Topologie pro směrování

Topologie pro směrování je obdobná jako pro firewall, jen je zde využito o jeden přepínač více a tedy i o 3 hosty více. Jak je zobrazeno v příloze A.2, každý z přepínačů má nastavenou IP adresu 192.168.n.1, kde **n** je pořadí přepínače. Přepínače se v tomto případě budou chovat jako L3 přepínače a postarají se o směrování dat v síti. Každá trojice hostů tvoří jednu síť, mají tedy přiřazenou IP adresu z rozsahu 192.168.n.m, kde **m** je pořadí hostů. Dále mají přiřazenou výchozí bránu na IP adresu přepínače, ke kterému jsou připojeni. Výchozí brána spolu s adresami přepínačů bude využita při směrování.

6.4 Hardwarové řešení

V této části budou uvedeny zařízení od jednotlivých firem, které lze použít na realizaci této problematiky. Níže je zobrazen soupis se zařízeními od jednotlivých firem. Vzhledem k tomu, že bylo nalezeno názornější řešení pomocí aplikace Mininet, tak byl vypuštěn přepínač, pomocí kterého měla být úloha realizována.

Tab. 6.2: Tabulka dostupných zařízení

Poskytovatel	**Zařízení**
Juniper Networks	EX4600, EX9200, MX80, MX240, MX480, MX960, MX2010, MX2020, QFX5100
Centec	V580, V350, V330, V150
HP	2920, 3500, 3800, 5406, 5412, 6600
Dell	N1500
Huawei	S12700 series, S7700 series
NoviFlow	100-000-401, 100-000-402, 100-000-501, 100-000-502, 100-000-505, 100-000-601, 100-000-602, 100-000-701, a další
Quanta	T5032-LY6, T3048-LY9A, T3048-LY8, T3048-LY9, T5032-LY6, T1048-LB9M, T1048-LB9
N3C	PF5240, PF5248, PF5340-48, PF5340-32, QX-S1000 Series, QX-S1000 Series a další
Alcatel	OmniSwitch 9900, 6450, 6860(E), 6900
Cisco	Nexus 3524-X/XL, Nexus 3548-X/XL, 34180YC, Nexus 3464C, Nexus 3432D-S, Nexus 3408-S, Nexus 3232C, Nexus 3264Q, Nexus 3264C-E, 3132C-Z, Nexus 3048

6.5 Návrh a simulace síťové topologie ve VMwaru

V této sekci bude podrobně popsán návrh reálné síťové topologie. Bude přiblížena konfigurace jednotlivých částí topologie, jmenovitě kontrolér Ryu, OpenVSwitch a softwarová komponenta Mininet.

6.5.1 Příprava virtuálního prostředí

Ve VMwaru budou vytvořeny dva virtuální stroje(VM). První s názvem Kontrolér, na které bude realizovaný kontrolér Ryu. Druhý s názvem Mininet, jenž bude realizovaný pro vytvoření tří topologií viz. 6.3, Tyto VM budou propojeny OpenVswitchem.

Příprava prostředí pro Ryu

Nejprve bude na virtuální stroj nainstalován operační systém Ubuntu 19.10. Následně mu budou přidělena síťová rozhraní NAT, LAN segment, který bude využit k propojení virtuálních strojů a rozhraní Host-only network, jenž bude sloužit k případnému spojení s přepínačem. Poté bude nainstalován potřebný software:

- net-tools - balíček obsahující nástroje pro síťovou konfiguraci.
- python-dev - balíček, který poskytuje hlavičkové soubory potřebné pro rozšíření pythonu.
- libffi - knihovna poskytující programovací rozhraní na vysoké úrovni pro různé konvence volání.
- libssl - část OpenSSL podporující TLS.
- libxml2 - XML C parser a sada nástrojů vyvinutá pro projekt Gnome. Implementuje standardy značkovacích jazyků.
- libxslt1 - obsahuje knihovnu XSLT C vyvinutá pro projekt GNOME. XSLT definuje transformaci souborů XML do jiného formátu (HTML, prostý text atd.)
- zlib1g - knihovna, jejíž hlavní účelem je komprese dat.

V další fázi bude naklonován repozitář z githubu, na které se nachází Ryu příkazem:

```
sudo git clone git://github.com/osrg/ryu.git
```

A v následujícím kroku bude nainstalován samotný kontrolér a Flow Manager příkazy:

```
sudo apt install ryu
git clone https://github.com/martimy/flowmanager
```

Posledním krokem bude trvalé nastavení IP adresy **10.100.107.10/24** na rozhraní ens192 (LAN segment) pomocí NetworkManagera. V základní konfiguraci má Ryu automaticky přidělenou IP adresu rozhraní virtuálního stroje a naslouchá na portu 6653, ale toto nastavení lze změnit.

Příprava a konfigurace prostředí pro Mininet

V prvé řadě bude na virtuální stroj nainstalován taktéž operační systém Ubuntu 19.10. Následně bude přiděleno virtuálnímu stroji rozhraní ens192 a nastaveno jako LAN segment, aby byl ve stejné síti jako kontrolér. Dalším krokem bude nainstalování **net-tools** a naklonování repozitáře z githubu příkazem:

```
sudo git clone https://github.com/openvswitch/ovs.git
```

a následná instalace pomocí příkazu:

```
sudo apt install openvswitch-switch
```

Poslední nainstalovaný program bude Mininet pomocí příkazu:

```
sudo apt install mininet.
```

Konfigurace OpenVSwitche

Po nainstalovaní OpenVSwitche bude následovat jeho nastavení. To bude realizováno při spuštění síťové topologie realizované Mininetem. Po spuštění topologie se OVS přidělí mosty s názvy jednotlivých přepínačů a každému z nich bude přiřazena adresa kontroléru. Jak již bylo zmíněno v teoretické části, tak kontrolér komunikuje s OpenFlow přepínačem pomocí šifrovaného TCP spojení. Z tohoto důvodu musí být OpenVswitchi přidělen kontrolér Ryu na adrese rozhraní virtuálního stroje obsahujícího kontrolér. Ryu bude následně komunikovat skrze toto spojení.

6.6 Simulace virtuálních topologií

V této části budou testovány jednotlivé komponenty síťové topologie a posléze bude provedena simulace komunikace napříč sítěmi, za pomoci aplikace Mininet.

6.6.1 Propojení Ryu a Mininetu

Na obrázku 6.5 je zobrazen výpis logu OpenVSwitche po propojení s kontrolérem. Výpis říká, že na OVS byl úspěšně vytvořen most s1 na portu 65334, kterému byla přiřazeny rozhraní z Mininetu, a to **s1-eth1-4,** na portech 1-4. Dále je z výpisu patrné, že most dostal identifikátor **datapath ID**. Další důležitou informací, která

je zobrazena na obrázku níže, je propojení s kontrolérem pomocí TCP na IP adrese **10.100.107.10:6653**. Poslední řádek říká, že byl přidán první zápis do tabulky toků.

```
2020-04-30T08:04:48.645Z|159090|bridge|INFO|bridge s1: added interface s1-eth2 on port 2
2020-04-30T08:04:48.646Z|159091|bridge|INFO|bridge s1: added interface s1-eth1 on port 1
2020-04-30T08:04:48.646Z|159092|bridge|INFO|bridge s1: added interface s1-eth4 on port 4
2020-04-30T08:04:48.646Z|159093|bridge|INFO|bridge s1: added interface s1-eth3 on port 3
2020-04-30T08:04:48.654Z|159094|bridge|INFO|bridge s1: added interface s1 on port 65534
2020-04-30T08:04:48.654Z|159095|bridge|INFO|bridge s1: using datapath ID 0000000000000001
2020-04-30T08:04:48.654Z|159096|connmgr|INFO|s1: added service controller "punix:/var/run/openvswitch/s1.mgmt"
2020-04-30T08:04:48.654Z|159097|rconn|INFO|s1<->tcp:10.100.107.10:6653: connecting...
2020-04-30T08:04:48.654Z|159098|connmgr|INFO|s1: added primary controller "tcp:10.100.107.10:6653"
2020-04-30T08:04:48.660Z|159099|rconn|INFO|s1<->tcp:10.100.107.10:6653: connected
2020-04-30T08:04:58.663Z|159100|connmgr|INFO|s1<->tcp:10.100.107.10:6653: 1 flow_mods 10 s ago (1 adds)
```

Obr. 6.5: Výpis logu OpenVswitche po spojení s kontrolérem

Obrázek 6.6 zobrazuje výpis základního toku (flow) pro most s1. Jsou v něm zobrazeny základní informace, ze kterých je patrné, že veškerý provoz je směrován na kontrolér (položka actions), jenž posléze vyhodnocuje provoz. Priorita s hodnotou 0 značí, že platí pro neznámý provoz a položka table říká, ve které tabulce se nachází.

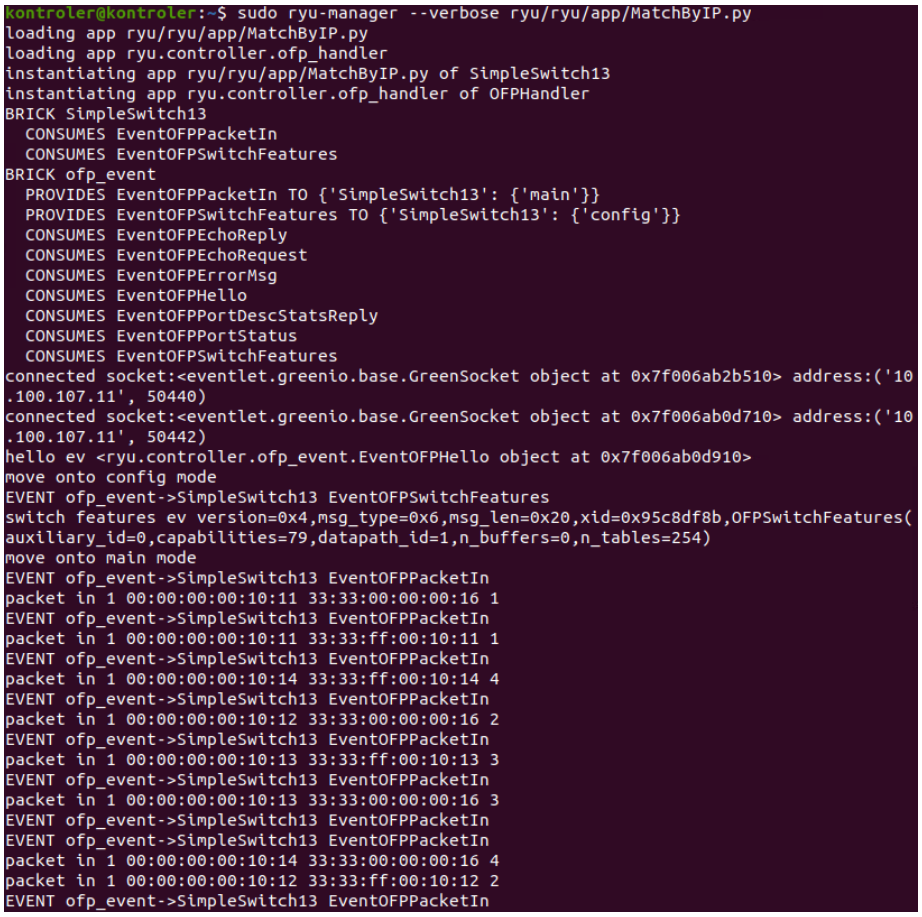
```
mininet@mininet:~$ sudo ovs-ofctl dump-flows s1
 cookie=0x0, duration=26.291s, table=0, n_packets=28, n_bytes=2344, priority=0
 actions=CONTROLLER:65535
```

Obr. 6.6: Výpis výchozího toku v tabulce toků

Přesuneme se na kontrolér Ryu, kde na obrázku 6.8 můžeme vidět spouštění skriptu **MatchByIP**, který využívá modul **SimpleSwitch13** a prvotní komunikaci Ryu s OpenVSwitchem. Je zde naznačeno přijímání packet-IN od přepínače, kde je vidět příchozí, odchozí IP adresa a číslo portu. Tyto pakety slouží ke komunikaci mezi přepínačem a kontrolérem. Dále, na obrázku níže, je zobrazena zpráva, kterou odesílá kontrolér přepínači při spuštění, a to nastavení základního toku, který byl popsán na obrázku 6.7. Jsou zde vidět identifikátory **datapathID**, které slouží k identifikaci trasy k přepínači a **auxiliaryID**, sloužící k určení typu připojení kontroléru a přepínače. Ostatní položky specifikují základní vlastnosti přepínače.

```
EVENT ofp_event->SimpleSwitch13 EventOFPSwitchFeatures
switch features ev version=0x4,msg_type=0x6,msg_len=0x20,xid=0xbfe7416b,OFPSwitchFeatures(
auxiliary_id=0,capabilities=79,datapath_id=1,n_buffers=0,n_tables=254)
```

Obr. 6.7: Odeslání toku na OVS



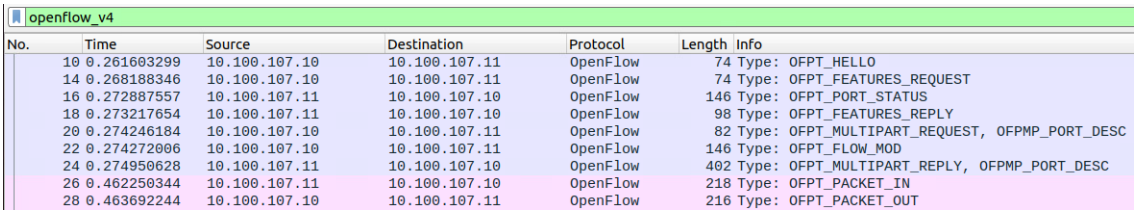
Obr. 6.8: Navázání komunikace kontroléru s OVS

6.6.2 Simulace komunikace pomocí Mininetu

V této sekci bude popsána simulace virtuální topologie a její výsledky. Pro realizaci bude použita topologie o 1 přepínači a 4 hostech, která je podrobněji popsána v sekci 6.3.1. Před spuštěním skriptu, sloužícího k sestavení topologie, bylo zapotřebí nejprve spustit kontrolér pomocí aplikace **MatchByIP** příkazem:

```
$ sudo ryu-manager ryu/ryu/app/MatchByIP
```

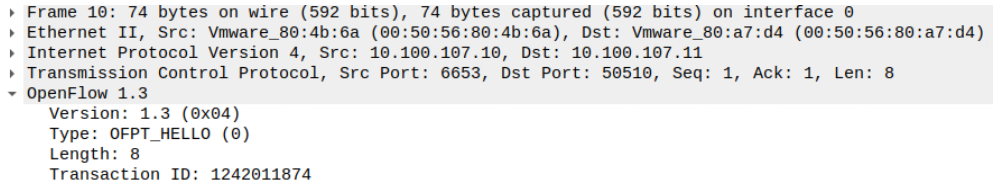
Pro simulaci byla upravena aplikace SimpleSwitch13, tak aby v jednotlivých tocích byla zapsána zdrojová a cílová IP adresa (aplikace s názvem **MatchByIP**). Jako zobrazovací nástroj průběhu simulace byl použit Wireshark, ve kterém se budou zaznamenávat zprávy mezi kontrolérem a přepínačem. Po spuštění kontroléru si s OVS vymění následující sérii paketů, která slouží k navázání komunikace.



Obr. 6.9: Navázání komunikace OVS s kontrolérem

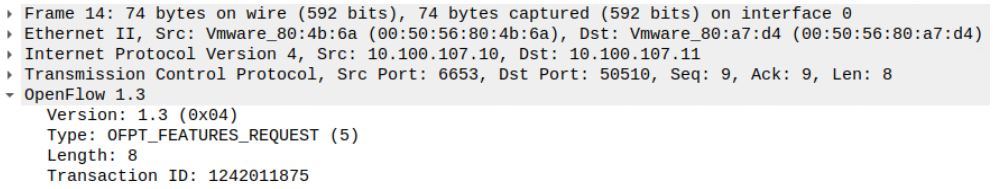
Navázání spojení

Komunikace začíná paketem **HELLO**, který je odesílán kontrolérem přepínači po ustálení spojení a domlouvají se jím, jakou verzi OpenFlow budou používat (viz. 6.10). V případě, že daný přepínač nepodporuje verzi, kterou kontrolér nabídnul, odešle paket **ERROR**.



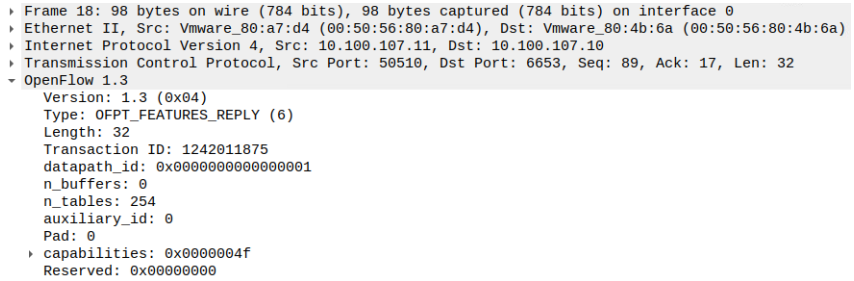
Obr. 6.10: HELLO paket

Po navázání a ustálení TCP spojení je odeslán **Feature Request** (viz. 6.11). Tento paket obsahuje pouze hlavičku, v níž se nachází pole **Feature Request**. Kontrolér se tím ptá přepínače jaké má vlastnosti.

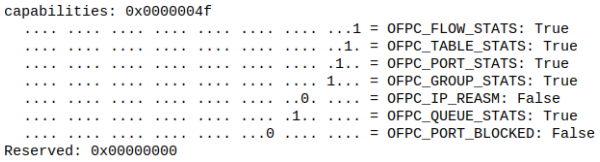


Obr. 6.11: Feature Request paket

Odpovědí od přepínače na tento paket je **Feature Reply** (viz. 6.12), který obsahuje vlastnosti daného přepínače. Mezi vlastnosti patří počet tabulek toků a jeho schopnosti (údaje podporované danou verzí protokolu OpenFlow). Vlastnosti tohoto přepínače jsou zobrazeny na obrázku 6.13. Zde můžeme vidět, že podporuje statistiky toků, tabulky portů, skupin a front. Podpora těchto vlastností je vyznačena logickou 1 u dané položky.

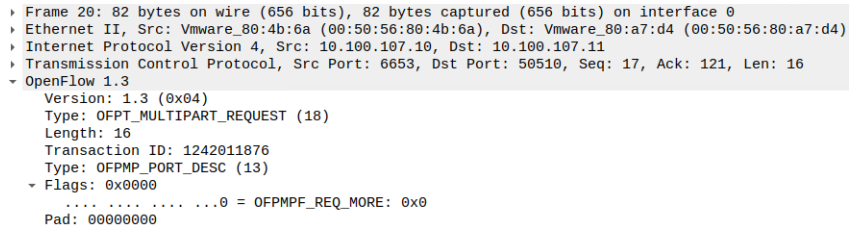


Obr. 6.12: Feature Reply paket



Obr. 6.13: Vlastnosti přepínače

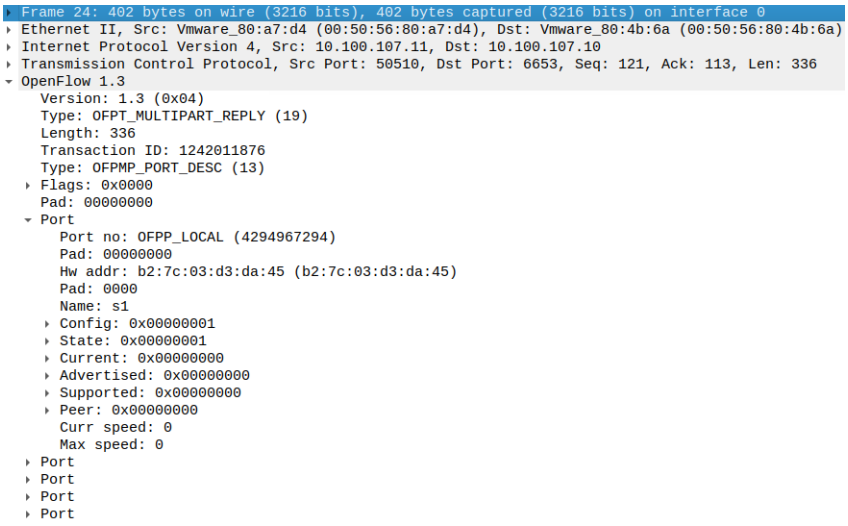
Dalším krokem, který kontrolér vykoná je odeslání paketu **OFPT Multipart Request** s typem **OFMP Port Desc** (viz. 6.14), jímž se dotazuje na parametry portů přepínače. Takto je paket pojmenován pouze u OpenFlow verze 1.3. U předchozích verzí může být vidět pod názvem Stats Request nebo Barrier Request.



Obr. 6.14: OFPT Multipart Request paket

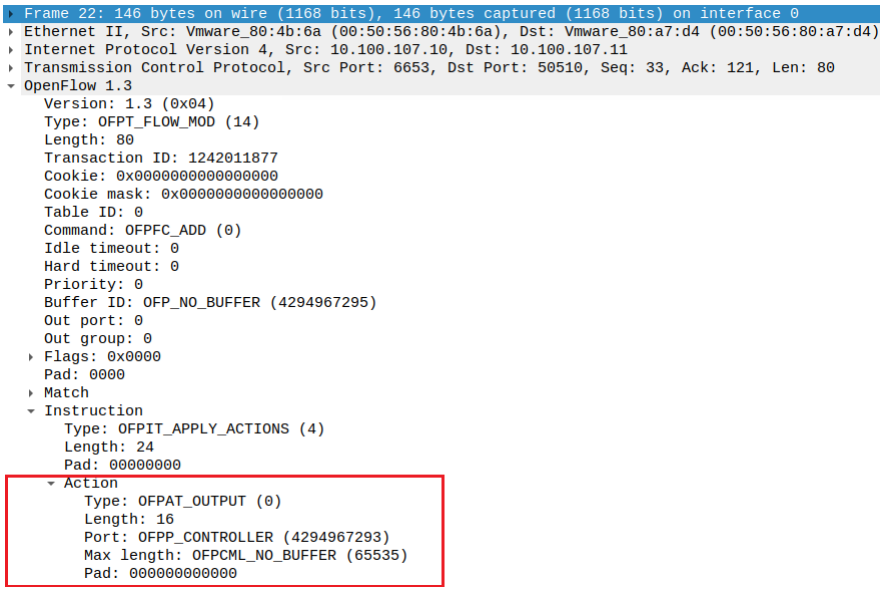
Odpovědí přepínače na předchozí paket je **OFPT Multipart Reply** (viz. 6.15), čímž přepínač odesílá vlastnosti jednotlivých portů kontroléru. Tento paket obsahuje kupříkladu název, informace o konfiguraci, stavu, maximální a současné rychlosti daného portu. Tyto informace kontrolér zaznamenává do jednotlivých toků.

Poslední věcí, kterou kontrolér při navázání spojení řeší, je odeslání paketu **FLOW MOD** (viz. 6.16). Ten slouží k zapsání prvního toku do tabulky. Jedná se o paket, který obsahuje informace o tom, jak s podobnou zprávou zacházet v případě, že přijde znovu. Nejdůležitější informací, kterou obsahuje je akce, kterou má



Obr. 6.15: OFPT Multipart Reply paket

vykonat v případě, že se takový paket znovu objeví. Je to akce **OFPP CONTROLLER**, která říká, že podobný paket má odeslat přímo kontroléru. Při prvotním startu kontroléru, kdy ještě nezná síť to znamená že, přepínač má přeposílat veškerou komunikaci na kontrolér. Dále obsahuje Table ID, které udává číslo tabulky, prioritu, jenž rozhoduje o váze daného toku (vyšší priorita znamená vyšší váhu) a další položky označující doplňkové informace k danému toku.



Obr. 6.16: FLOW MOD paket

Test Komunikace

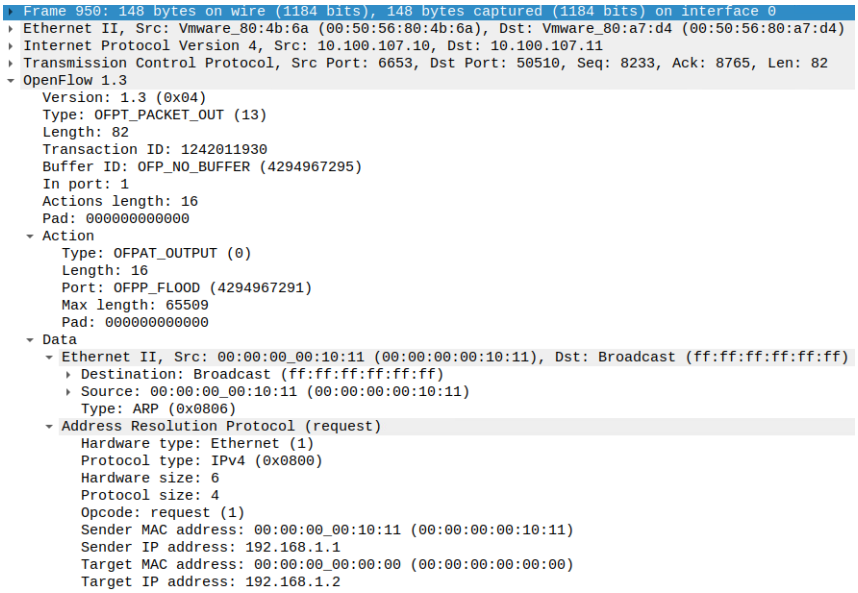
Po úspěšném navázání spojení byla otestována komunikace mezi H1 a H2 prostřednictvím ICMP zprávy. První akce, kterou vykoná přepínač při přijetí požadavku na přeposlání ICMP zprávy od H1, je odeslání zprávy Packet-IN kontroléru. Ta je zobrazena na obrázku 6.17). Jsou zde znázorněny údaje o paketu (zdrojová adresa, cílová adresa, bufferID atd.) a jedna z nejdůležitějších informací - položka **Reason**, která sděluje proč byl paket zadržen a přeposlán kontroléru. V tomto případě je tam vložena zpráva **NO MATCH**, jenž říká, že pro daný paket nemá přepínač uložený datový tok v tabulce.

```
▸ Frame 949: 150 bytes on wire (1200 bits), 150 bytes captured (1200 bits) on interface 0
▸ Ethernet II, Src: Vmware_80:a7:d4 (00:50:56:80:a7:d4), Dst: Vmware_80:4b:6a (00:50:56:80:4b:6a)
▸ Internet Protocol Version 4, Src: 10.100.107.11, Dst: 10.100.107.10
▸ Transmission Control Protocol, Src Port: 50510, Dst Port: 6653, Seq: 8681, Ack: 8233, Len: 84
▾ OpenFlow 1.3
    Version: 1.3 (0x04)
    Type: OFPT_PACKET_IN (10)
    Length: 84
    Transaction ID: 0
    Buffer ID: OFP_NO_BUFFER (4294967295)
    Total length: 42
    Reason: OFPR_NO_MATCH (0)
    Table ID: 0
    Cookie: 0x0000000000000000
  ▾ Match
      Type: OFPMT_OXM (1)
      Length: 12
    ▸ OXM field
      Pad: 00000000
    Pad: 0000
  ▾ Data
    ▾ Ethernet II, Src: 00:00:00_00:10:11 (00:00:00:00:10:11), Dst: Broadcast (ff:ff:ff:ff:ff:ff)
      ▸ Destination: Broadcast (ff:ff:ff:ff:ff:ff)
      ▸ Source: 00:00:00_00:10:11 (00:00:00:00:10:11)
        Type: ARP (0x0806)
    ▾ Address Resolution Protocol (request)
        Hardware type: Ethernet (1)
        Protocol type: IPv4 (0x0800)
        Hardware size: 6
        Protocol size: 4
        Opcode: request (1)
        Sender MAC address: 00:00:00_00:10:11 (00:00:00:00:10:11)
        Sender IP address: 192.168.1.1
        Target MAC address: 00:00:00_00:00:00 (00:00:00:00:00:00)
        Target IP address: 192.168.1.2
```

Obr. 6.17: Packet In odeslaný přepínačem při prvním kontaktu s paketem

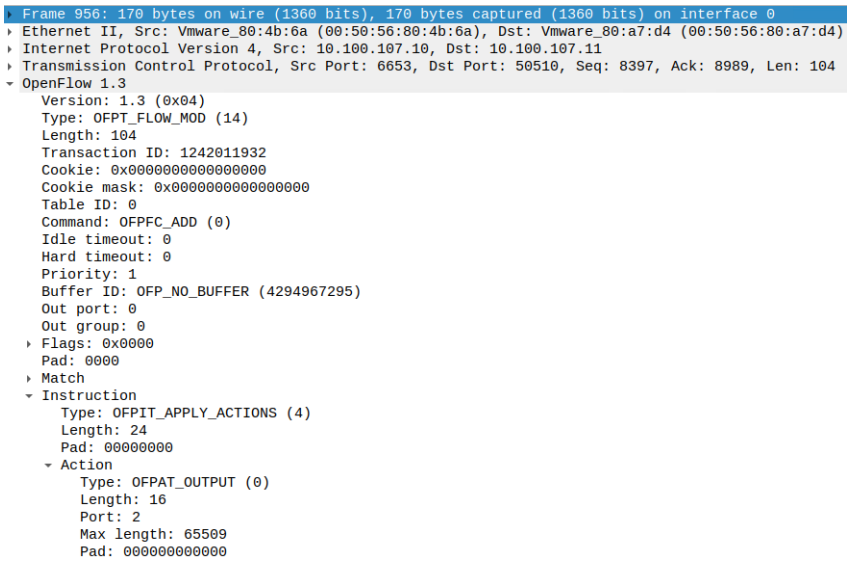
Odpovědí od kontroléru je zpráva Packet-Out (viz. 6.18), která obsahuje, mimo informací o daném paketu, položku **action**, jenž říká co má přepínač s daným paketem dělat. V tomto případě obsahuje zprávu **OFPT FLOOD**, čímž vzkazuje přepínači, že má odeslat zprávy formou záplavy (odeslání na všechny rozhraní).

Výsledkem této zprávy je odpověď hledané stanice. Ta je odeslána na kontrolér formou zprávy Packet-In, jenž vypadá obdobně jako předchozí. Kontrolér na tuto zprávu odpoví paketem **FLOW MOD** (viz. 6.19). Ten obsahuje položku Instruction, která je typu **OFPIT APPLY ACTIONS** a obsauje parameter **Action**, jenž je typu **OFPAT OUTPUT**. Prostřednictvím těchto informací přikazuje kontrolér přepínači uložení daného datového toku s informacemi o zdrojové, cílové ad-



Obr. 6.18: Packet Out se zprávou FLOOD

rese a portu, na nějž má podobné pakety odesílat. To znamená, že pokud se objeví podobný paket znovu, bude odeslán na port 1. Zároveň s **FLOW MOD** odesílá kontrolér i Packet-Out, kterým přikazuje přeposlání ICMP na H2. Ta odpoví formou ICMP reply a jelikož neví kam odeslat paket od stanice H2, celý koloběh se opakuje, dokud H1 nepřijme ICMP reply.



Obr. 6.19: FLOW MOD paket

Po dokončení této části budou mít studenti za úkol seznámit se s aplikací Flow-

manager, která slouží ke konfiguraci datových toků na jednotlivých přepínačích. V tomto případě bude využita k realizaci pravidel ACL. K počáteční konfiguraci je zde použit YANG model (viz. příloha A.4), který rozdělí datové toky do tří tabulek. První slouží jako vstupní, její funkce bude pouze odeslat veškerý provoz do druhé tabulky. Ve druhé jsou nadefinovány pravidla ACL a to tak, že komunikace z PC1 na PC3 a z PC2 na PC4 bude zakázána a ostatní provoz bude přeposlán do třetí tabulky. Ta má na starost odesílání veškerého příchozího provozu na výstup a současně odesílá neznámé pakety kontroléru. Tomuto řešení se říká zřetězení, nebo-li pipelining. Studenti budou mít posléze za úkol nakonfigurovat některé datové toky.

6.7 Firewall

Jednou z nejzákladnějších konfigurací, kterou je nutné udělat, je nastavení firewallu. Tato konfigurace nastaví přístupy z jednotlivých zařízení na ostatní. Firewall lze nastavit pomocí modulu, který Ryu poskytuje. Níže bude popsána konfigurace firewallu a následně provedena simulace topologie vytvořené v prostředí Mininet, která je popsána v sekci 6.3.1.

6.7.1 Konfigurace a simulace

Konfigurace bude realizována pomocí modulu Firewall, který poskytuje možnost konfigurace pravidel firewallu. Pro zobrazení výpisu, jenž je vidět na obrázku(6.20) je zapotřebí provést spuštění příkazem:

```
$ ryu-manager ryu/ryu/app/rest_firewall.py
```

```
kontroler@kontroler:~$ sudo ryu-manager ryu/ryu/app/rest_firewall.py
[sudo] password for kontroler:
loading app ryu/ryu/app/rest_firewall.py
loading app ryu.controller.ofp_handler
instantiating app None of DPSet
creating context dpset
creating context wsgi
instantiating app ryu/ryu/app/rest_firewall.py of RestFirewallAPI
instantiating app ryu.controller.ofp_handler of OFPHandler
(25494) wsgi starting up on http://0.0.0.0:8080
[FW][INFO] dpid=0000000000000001: Join as firewall.
[FW][INFO] dpid=0000000000000002: Join as firewall.
[FW][INFO] dpid=0000000000000003: Join as firewall.
```

Obr. 6.20: Spuštění Firewallu

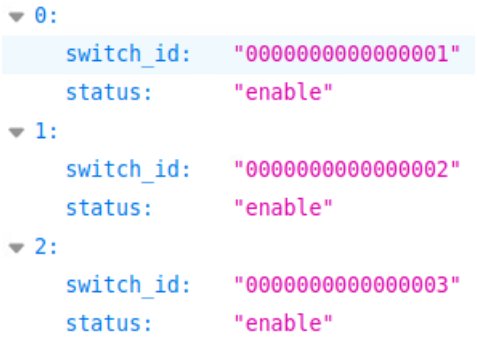
Po spuštění tohoto modulu je zapotřebí spustit správnou topologii v Mininetu. To je vykonáno pomocí příkazu:

```
$ sudo ./firewallTopology
```

Následně je nutné povolit firewall na přepínačích, kde se bude používat. Toto nastavení se dělá příkazem:

```
$ curl -X PUT
http://localhost:8080/firewall/module/enable/(switchID)
```

Nebo je možné povolení na všech přepínačích, kdy se **switchID** nahradí textem **all**. Toto nastavení je zobrazeno v grafickém rozhraní kontroléru na obrázku 6.21. Do grafického rozhraní firewallu, kde je zobrazeno nastavení v čitelné podobě, se lze dostat zadáním IP adresy kontroléru s portem 8080 a cesty /firewall/module/status (stav firewallu) nebo /firewall/module/rules/switchID pro zobrazení nastavení firewallu daného přepínače.



Obr. 6.21: Povolení Firewallu

Dalším krokem je nadefinování funkčnosti daného firewallu, kde je z důvodu testování použito následující série příkazů, která povolí pouze komunikaci protokolem ICMP mezi PC1 a PC2. Zadání příkazu musí být oboustranné, aby bylo zajištěno odeslání odpovědi. Příkaz **curl** slouží k odeslání informací na server, nebo jejich přijetí kde, **-X** značí používání proxy serveru, **POST** znamená odesílání dat. za **-d** následuje zápis dat ve formátu json. V zápisu je udána nejprve zdrojová adresa **nw_src**, cílová adresa **nw_dst** a protokol **nw_proto**, který chceme povolit nebo zakázat. Mezi volitelné položky patří akce (actions). Tu zadáváme pokud je nutné specifikovat zakázání, nebo povolení daného protokolu (v základu je nastaven jako povoleno), dále pak priorita, jenž udává v jakém pořadí bude nahlíženo na jednotlivá pravidla.

```
$ curl -X POST -d '{"nw_src": "192.168.1.1",
"nw_dst": "192.168.1.2", "nw_proto": "ICMP"}'
http://localhost:8080/firewall/rules/0000000000000001
```

```
$ curl -X POST -d '{"nw_src": "192.168.1.2",
"nw_dst": "192.168.1.1", "nw_proto": "ICMP"}'
http://localhost:8080/firewall/rules/0000000000000001
```

Po nastavení firewallu se toky pro most S1 uložily jak do tabulky toků v OVS, které jsou zobrazeny na obrázku 6.22, tak do grafického rozhraní, které je možné vidět na obrázku 6.23, kde ve vrchní části je **switchID** přepínače, jenž byl konfigurován. Níže je zobrazena nastavená konfigurace. Zde můžeme vidět, že **rule_id** prvního pravidla bylo na nastaveno na 1, priorita také dostala hodnotu 1, protože nebylo řečeno jinak. Dále je zde vidět typ sítě, zdrojová a cílová adresa, protokol, kterého se to týká a co se má provést (akce).

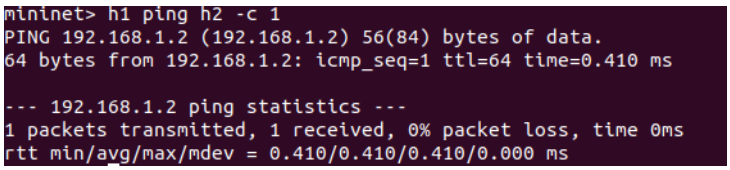
```
mininet@mininet:~$ sudo ovs-ofctl dump-flows s1
[sudo] password for mininet:
 cookie=0x0, duration=749.181s, table=0, n_packets=4, n_bytes=168, priority=65534,
arp actions=NORMAL
 cookie=0x1, duration=307.579s, table=0, n_packets=1, n_bytes=98, priority=1,icmp,
nw_src=192.168.1.1,nw_dst=192.168.1.2 actions=NORMAL
 cookie=0x2, duration=293.047s, table=0, n_packets=1, n_bytes=98, priority=1,icmp,
nw_src=192.168.1.2,nw_dst=192.168.1.1 actions=NORMAL
 cookie=0x0, duration=749.181s, table=0, n_packets=8, n_bytes=826, priority=0 acti
ons=CONTROLLER:128
```

Obr. 6.22: Tabulka toků S1

Obr. 6.23: Tabulka toků S1

Funkčnost lze ověřit použitím příkazu ping z PC1 na PC2, který používá protokol ICMP. Na obrázku 6.24 je ověření, že zpráva ICMP byla úspěšně odeslána a dostala

odpověď zpět. Pomocí modulu firewall je možné nastavovat spoustu různých pra-



Obr. 6.24: Simulace ICMP

videl. Ukázka výše byla použita pouze jako testovací. Lze například omezit provoz pro další protokoly (UDP a TCP). Tento modul podporuje jak protokol IPv4, tak IPv6. Některá z dalších pravidel budou mít za úkol natavit studenti viz. Laboratorní návod v příloze B.

6.8 Směrování prostřednictvím Ryu

Pomocí kontroléru lze nastavit směrování provozu jednotlivých zařízení. Pro toto nastavení je využito modulu Router. Níže bude popsána konfigurace směrování s použitím Ryu a následná simulace prostřednictvím topologie vytvořené v Mininetu.

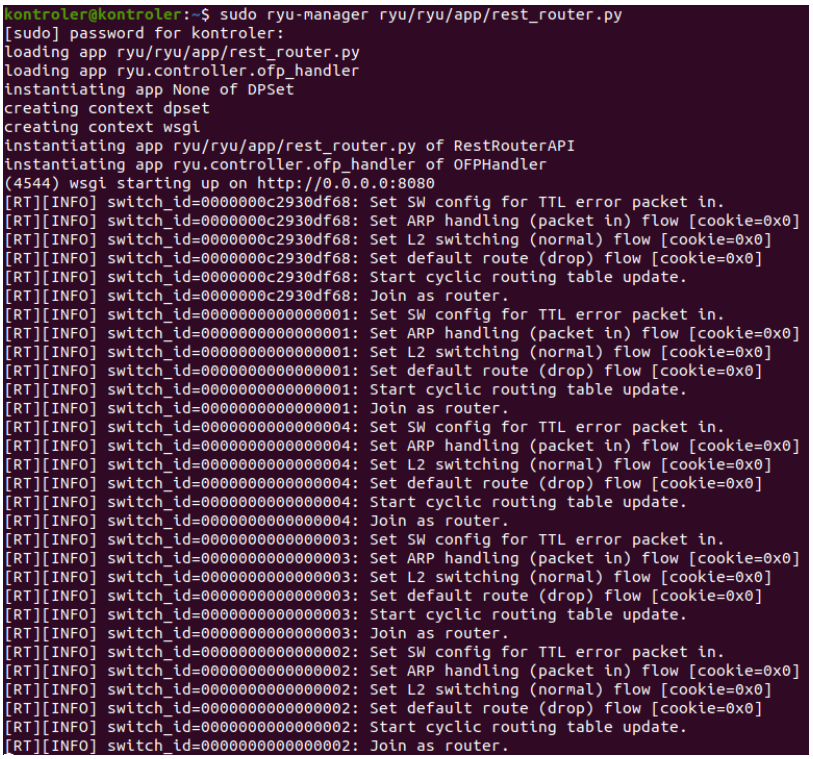
6.8.1 Konfigurace a simulace

Nastavení směrování na jednotlivých L3 přepínačích bude realizováno za pomoci modulu Router. Prvním krokem je spuštění Ryu, správného modulu a topologie v Mininetu následujícími příkazy:

```
$ ryu-manager ryu/ryu/app/rest_router.py
$ sudo ./routerTopo
```

Po použití těchto příkazů se zobrazí výpis na 6.25. Tento výpis říká, že všechny přepínače byli úspěšně připojeny ve funkci L3 přepínače a je možné u nich provádět směrování. Dále bude popsán princip konfigurace jednotlivých zařízení. Celou konfiguraci budou realizovat studenti pomocí laboratorního návodu.

Po úspěšném startu všech L3 přepínačů je zapotřebí každému z nich nastavit IP adresy, které se používají v sítích, jenž přepínač obsluhuje jako brány a adresy rozhraní, jimiž komunikuje s ostatními přepínači. Kupříkladu jak je zobrazeno na obr. 6.4, přepínač S1 používá dvě takovéto adresy. První z nich je brána 192.168.1.1/24, ke které náleží síť 192.168.1.0/24. Jedná se o síť, ve které se nachází PC připojené k tomuto přepínači. Druhá je IP adresa rozhraní 192.168.10.10, jenž je použita ke komunikaci s přepínačem S2. Ke konfiguraci těchto IP adres je použito následujícího příkazu:



Obr. 6.25: Úvodní výpis modulu Router

```
$ curl -X POST -d '{"address":"(IP adresa)/(maska)"}'
http://localhost:8080/router/(switchID)/(vlan)
```

V příkazu je pouze jedna důležitá položka a to položka address, za kterou se doplní IP adresa rozhraní, či brány a maska sítě. Například pro přidání adres **192.168.1.1** a **192.168.10.10** přepínači S1 a adres **192.168.2.1** a **192.168.10.1** na přepínači S2 budou příkazy vypadat následovně:

```
$ curl -X POST -d '{"address":"192.168.1.1/24"}'
http://localhost:8080/router/0000000000000001

$ curl -X POST -d '{"address":"192.168.10.10/24"}'
http://localhost:8080/router/0000000000000001
```

a pro pro přidání adres 192.168.2.1 a 192.168.10.1 na přepínač S2 pomocí:

```
$ curl -X POST -d '{"address":"192.168.2.1/24"}'
http://localhost:8080/router/0000000000000002

$ curl -X POST -d '{"address":"192.168.10.1/24"}'
http://localhost:8080/router/0000000000000002
```

Dalším krokem bude nastavení výchozích rout, kterými bude případně odesílán provoz. Toto nastavení se používá pouze na kraji sítě, kde se nachází jeden port vedoucí mimo síť. Zde se nastavuje pouze přidělením brány za parametr gateway, kterou bude provoz směrován. Konfigurace se provede příkazem:

```
curl -X POST -d '{"gateway":"(IP adresa)"}'
http://localhost:8080/router/(switchID)/(vlan)
```

Kupříkladu pro přepínač S1 a S2 je pro konfiguraci výchozí routy použito následujících příkazů:

```
curl -X POST -d '{"gateway":"192.168.10.1"}'
http://localhost:8080/router/0000000000000001

curl -X POST -d '{"gateway":"192.168.10.10"}'
http://localhost:8080/router/0000000000000002
```

Posledním krokem bude nastavení statické routy. V tomto případě pro S1 ani S2 není zapotřebí nastavovat tuto routu, protože již příkazy výše byla umožněna komunikace mezi sítěmi 192.168.1.0 a 192.168.2.0, což jsou právě sítě přepínače S1 a S2. Případná konfigurace statické routy by byla provedena pomocí:

```
curl -X POST -d '{"destination":"(IP adresa)/(maska)",
"gateway":"(IP adresa)"}'
http://localhost:8080/router/(switchID)/(vlan)
```

Tento příkaz obsahuje dva parametry. První z nich je destination, jenž je cílovou adresu sítě a druhý parametr je gateway, který udává bránu využitou ke vstupu do sítě. Zápis by mohl vypadat následnovně:

```
curl -X POST -d '{"destination":"192.168.1.0/24",
"gateway":"192.168.10.10"}'
http://localhost:8080/router/0000000000000002
```

Po výše prováděné konfiguraci už je možné komunikovat mezi sítí přepínače S1 a S2. Níže na obrázcích je zobrazena směrovací tabulka jednotlivých L3 přepínačů S1 a S2 v grafickém rozhraní Ryu. Jsou zde vidět jednotlivé adresy, jejich ID a routy, které byli přiřazeny, v tomto případě pouze default route a její ID. Na posledních dvou obrázcích jsou tabulky toků S1 a S2. Je zde vidět přiřazení jednotlivých adres a další informace o daném toku. Vzhledem k tomu, že byl proveden test pomocí ICMP zprávy, je zde zobrazen datový tok rozhraní odkud byla zpráva odeslána.

Route table S1

```
internal_network:
  0:
    address:
      0:
        address_id:   1
        address:      "192.168.1.1/24"
      1:
        address_id:   2
        address:      "192.168.10.10/24"
    route:
      0:
        route_id:     1
        destination:  "0.0.0.0/0"
        gateway:      "192.168.10.1"
```

Route table S2

```
internal_network:
  0:
    address:
      0:
        address_id:   1
        address:      "192.168.10.1/24"
      1:
        address_id:   2
        address:      "192.168.2.1/24"
    route:
      0:
        route_id:     1
        destination:  "0.0.0.0/0"
        gateway:      "192.168.10.10"
```

Obr. 6.26: Směrovací tabulka S1 a S2

```
mininet@mininet:~$ sudo ovs-ofctl dump-flows s1
 cookie=0x1, duration=1951.453s, table=0, n_packets=0, n_bytes=0, priority=1037,ip,nw_ds
t=192.168.1.1 actions=CONTROLLER:65535
 cookie=0x2, duration=1931.648s, table=0, n_packets=0, n_bytes=0, priority=1037,ip,nw_ds
t=192.168.10.10 actions=CONTROLLER:65535
 cookie=0x1, duration=1767.476s, table=0, n_packets=0, n_bytes=0, idle_timeout=1800, pri
ority=35,ip,nw_dst=192.168.1.3 actions=dec_ttl,mod_dl_src:0a:97:4e:43:90:a4,mod_dl_dst:0
0:00:00:00:01:02,output:"s1-eth4"
 cookie=0x1, duration=1762.706s, table=0, n_packets=6, n_bytes=588, idle_timeout=1800, p
riority=35,ip,nw_dst=192.168.1.2 actions=dec_ttl,mod_dl_src:06:aa:04:b1:67:ad,mod_dl_dst
:00:00:00:00:01:01,output:"s1-eth3"
 cookie=0x1, duration=1951.453s, table=0, n_packets=6, n_bytes=588, priority=36,ip,nw_sr
c=192.168.1.0/24,nw_dst=192.168.1.0/24 actions=NORMAL
 cookie=0x2, duration=1931.648s, table=0, n_packets=0, n_bytes=0, priority=36,ip,nw_src=
192.168.10.0/24,nw_dst=192.168.10.0/24 actions=NORMAL
 cookie=0x1, duration=1951.453s, table=0, n_packets=0, n_bytes=0, priority=2,ip,nw_dst=1
92.168.1.0/24 actions=CONTROLLER:65535
 cookie=0x2, duration=1931.648s, table=0, n_packets=0, n_bytes=0, priority=2,ip,nw_dst=1
92.168.10.0/24 actions=CONTROLLER:65535
 cookie=0x0, duration=2270.494s, table=0, n_packets=14, n_bytes=696, priority=1,arp acti
ons=CONTROLLER:65535
 cookie=0x10000, duration=1790.760s, table=0, n_packets=8, n_bytes=784, priority=1,ip ac
tions=dec_ttl,mod_dl_src:9e:07:58:3f:d7:7d,mod_dl_dst:06:89:91:7f:e8:9d,output:"s1-eth1"
 cookie=0x0, duration=2270.494s, table=0, n_packets=232, n_bytes=25378, priority=0 actio
ns=NORMAL
```

Obr. 6.27: Tabulka toků S1

```
mininet@mininet:~$ sudo ovs-ofctl dump-flows s2
 cookie=0x1, duration=2047.066s, table=0, n_packets=0, n_bytes=0, priority=1037,ip,nw_ds
t=192.168.10.1 actions=CONTROLLER:65535
 cookie=0x2, duration=2038.917s, table=0, n_packets=0, n_bytes=0, priority=1037,ip,nw_ds
t=192.168.2.1 actions=CONTROLLER:65535
 cookie=0x1, duration=2047.066s, table=0, n_packets=0, n_bytes=0, priority=36,ip,nw_src=
192.168.10.0/24,nw_dst=192.168.10.0/24 actions=NORMAL
 cookie=0x2, duration=2038.916s, table=0, n_packets=0, n_bytes=0, priority=36,ip,nw_src=
192.168.2.0/24,nw_dst=192.168.2.0/24 actions=NORMAL
 cookie=0x1, duration=2047.066s, table=0, n_packets=0, n_bytes=0, priority=2,ip,nw_dst=1
92.168.10.0/24 actions=CONTROLLER:65535
 cookie=0x2, duration=2038.917s, table=0, n_packets=3, n_bytes=294, priority=2,ip,nw_dst
=192.168.2.0/24 actions=CONTROLLER:65535
 cookie=0x0, duration=2448.460s, table=0, n_packets=13, n_bytes=654, priority=1,arp acti
ons=CONTROLLER:65535
 cookie=0x10000, duration=1984.777s, table=0, n_packets=8, n_bytes=784, priority=1,ip ac
tions=dec_ttl,mod_dl_src:06:89:91:7f:e8:9d,mod_dl_dst:9e:07:58:3f:d7:7d,output:"s2-eth1"
 cookie=0x0, duration=2448.460s, table=0, n_packets=235, n_bytes=25588, priority=0 actio
ns=NORMAL
```

Obr. 6.28: Tabulka toků S2

Závěr

Cílem bakalářské práce **SDN pro řízení sítí LAN** bylo shromáždit informace o problematice Softwarově definovaných sítí, vytvořit topologii laboratorní úlohy, vybrat vhodný kontrolér, sepsat dostupná zařízení a vytvořit laboratorní návod, který studenti budou využívat pro vypracování laboratorní úlohy.

V jednotlivých kapitolách teoretické části byly sesbírány informace o architektuře **SDN**: popis jednotlivých vrstev, princip fungovaní přenosu dat a bezpečnost. Na konci tohoto odvětví byly shrnuty výhody a nevýhody této architektury. Další odvětví teoretické části bylo zaměřeno na **OpenFlow protokol**, zejména jeho princip a detailní rozbor jednotlivých částí. V neposlední řadě byl udělán soupis jednotlivých kontrolérů včetně stručného popisu, který sloužil k jejich porovnání. Další část byla zaměřena na protokol **NETCONF** a modelovací jazyk **YANG**, který s tímto protokolem úzce souvisí.

Praktická část byla věnována výběru vhodného kontroléru a jeho detailnějšímu popisu, zejména jeho architektury a důležitých modulů. Mezi existujícími kontroléry vybrán Ryu, protože obsahuje vhodné moduly, které budou využity k realizaci laboratorní úlohy a lze pomocí něj vysvětlit funkčnost SDN. Další výhodou oproti ostatním kontrolérům je dostupnost rozsáhlé dokumentace a kompatibilita s operačním systémem Linux.

Následujícím tématem praktické části bylo navrhnutí síťové topologie a popis principu fungování. Z důvodu názornosti bylo oproti jedné topologii s reálným přepínačem vytvořeno více topologií v aplikaci Mininet. Jmenovitě se jedná o původní topologii s jedním přepínačem, na které se studenti seznámí s technologií SDN. Následuje topologie vytvořená speciálně pro funkčnost firewallu, kde se nachází 3 přepínače a každý z nich obsluhuje 3 PC. V poslední řadě se jedná o topologii, která bude sloužit k nastavení statického směrování na L3 přepínačích. Ta sestává ze 4 přepínačů, kde každý má nastaven specifickou IP adresu sítě. Všechny přepínače v dříve zmíněných topologiích spravuje kontrolér Ryu. U topologie pro nastavení pravidel Firewallu, se kontrolér Ryu stará o konfiguraci jednotlivých pravidel a o jejich dodržování. U topologie sloužící k nastavení statického směrování se stará o jejich nastavení a obsluhu. V další řadě byla vytvořena tabulka dostupných zařízení, která zde slouží ve výsledku pouze jako informativní, aby zobrazila přehled dostupných zařízení. V případě návaznosti na další práci, bude tato tabulka použita k výběru správného přepínače.

Aby mohla být úloha realizována, muselo být připraveno pracoviště. Byly vytvořeny dva virtuální stroje. Jeden z nich s názvem Kontrolér, na kterém běží OS Linux, byl využit k umístění kontroléru Ryu a nástrojů potřebných k jeho běhu. Následně bylo připraveno síťové rozhraní, které slouží ke komunikaci s druhým virtuálním

strojem. Druhá VM, která byla vytvořena také s OS Linux, skládající se z aplikace Mininet, která slouží k realizaci topologií, virtuálního přepínače OVS a jeho nástrojů. OVS propojuje jednotlivé přepínače s kontrolérem Ryu. Následně i na této VM bylo nakonfigurováno rozhraní, sloužící ke vzájemné komunikaci.

V dalším kroku byla provedena simulace komunikace mezi těmito virtuálními stroji, která proběhla úspěšně. Následně byla otestována funkčnost všech topologií a spolupráce s moduly Router (směrování), Firewall (pravidla firewallu) a aplikace SimpleSwitch (jednoduchý přepínač), která taktéž prošla bez problémů. Po této simulaci byl vytvořen YANG model, který bude sloužit ke konfiguraci datových toků. Tento model byl odzkoušen a vyladěn. Jak již bylo zmíněno v praktické části, slouží k počáteční konfiguraci, kterou studenti využijí na konci prvního úkolu laboratorního návodu.

V poslední řadě byl realizován laboratorní návod, který má studentům pomoci s realizací laboratorní úlohy. Tento návod lze nalézt v příloze B. Úkolem studentů v této úloze bude seznámit se s technologií Softwarově definovaných sítí a používaným softwarem, jmenovitě s aplikací Flow manager, Ryu, Mininet a OVS. Dále pak nastavit jednoduchý Firewall, na kterém si vyzkouší konfiguraci pravidel firewallu pomocí Ryu. A v poslední řadě nastavení směrování pomocí kontroléru Ryu a modulu Router, kde se seznámí se základy statického směrování, za pomoci kontroléru a protokolu OpenFlow. Laboratorní návod je rozdělen na dvě samostatné úlohy. Jedna z nich bude sloužit k seznámení s technologií a druhá k realizaci firewallu a směrování.

V závěru této práce bych rád porovnal typickou lokální síť oproti SDN a změny oproti technologii NFV. Jak je patrné SDN je výhodnější oproti typické lokální síti ve všech ohledech. Mezi výhody patří konfigurace a správa celé sítě z jednoho místa, možnost řízení provozu na jednotlivých přepínačích, konfigurace QoS a nižší namáhání síťových zařízení. Jedinou nevýhodou, která byla během realizace této práce zjištěna je bezpečnost. Jak již bylo zmíněno kontrolér je mozkem celé sítě a shromažďují se v něm četná data. Právě proto se případný útočník zaměří na tento bod síťové infrastruktury. Tomuto lze zabránit zvolením vhodných bezpečnostních opatření.

Další možností virtualizace sítě je NFV (Network Functions Virtualization). Jak SDN, tak NFV využívají síťové abstrakce. Základním rozdílem je, že SDN se snaží o rozdělení funkce řízení sítě a přeposílání dat, zatímco NFV se snaží oddělit síťové funkce od hardwaru. Obě tyto technologie lze použít dohromady. Problematika této technologie je obsáhlá, složitá a není předmětem této práce.

Literatura

[1] What is virtualization? 2019. Opensource [online]. PO Box 1866 Mountain View, CA 94042 USA: Creative commons. [cit. 2019 27-10] <Dostupnéz:https://opensource.com/resources/virtualization>

[2] PANDEY, Vijoy. 2013. *The SDN Dial Tone – What Network Virtualization Should Do.* Sdxcentral [online]. [cit. 2019 28-10] Dostupné z: <https://www.sdxcentral.com/articles/contributed/network-virtualization-and-the-sdn-dial-tone/2013/07/

[3] *VIRTUALIZATION: What is virtualization?* 2018. Redhat [online]. Red Hat. [cit. 2019 30-10] Dostupné z: <https://www.redhat.com/en/topics/virtualization/what-is-virtualization>

[4] *Virtualization.* 2019. Ibm [online]. IBM Cloud Education. [cit. 2019 31-10] Dostupné z: <https://www.ibm.com/cloud/learn/virtualization-a-complete-guide>

[5] FEAMSTER, Nick, Jennifer REXFORD and Ellen ZEGURA. 2014. *The Road to SDN: An Intellectual History of Programmable Networks.* ACM SIGCOMM Computer Communication Review. 2014(44), 87-98.

[6] NADEAU, Thomas D. a Kenneth GRAY. *SDN: software defined networks.* Beijing: O'Reilly, [2013]. ISBN 978-1-449-34230-2.

[7] LESSING, Marlese. *Understanding the SDN Architecture - SDN Control Plane & SDN Data Plane.* Sdxcentral [online]. [cit. 2019 10-10] Dostupné z: <https://www.sdxcentral.com/networking/sdn/definitions/inside-sdn-architecture/>

[8] ARORA, Himanshu and Tarun THAKUR. *Software Defined Networking (SDN) - Architecture and role of OpenFlow.* Howtoforge [online].[cit. 2019 10-10] Dostupné z: <https://www.howtoforge.com/tutorial/software-defined-networking-sdn-architecture-and-role-of-openflow/>

[9] SHIN, Myung-Ki, Ki-Hyuk NAM and Hyoung-Jun KIM. *Softwarově definované sítě (SDN): Referenční architektura a otevřená API.* IEEE [online]. Jeju Island, South Korea. [cit. 2019-10-16] Dostupné z: <https://ieeexplore.ieee.org/document/6386859>

[10] KUBICA, Tomáš. 2014. *Malá učebnice OpenFlow (2) – úvod do OpenFlow 2014.* Netsvet [online]. [cit. 2019 10-13] Dostupné z: <https://www.netsvet.cz/mala-ucebnice-openflow-2-uvod-do-openflow/>

[11] SALISBURY, BRENT. *OpenFlow: Proactive vs Reactive Flows*. Networkstatic [online]. [cit. 2019 10-14] Dostupné z: <http://networkstatic.net/openflow-proactive-vs-reactive-flows/>

[12] SALMAN, Ola, Imad H. ETHAJJ, Ayman KAYSSI and Ali CHEBAB. 2016. *SDN controllers: A comparative study*. Mediterranean Electrotechnical Conference (MELECON) [online]. Lemesos, Cyprus: IEEE, 2016(18), 1-6. [cit 2019 11-18] Dostupné z: <https://ieeexplore.ieee.org/abstract/document/7495430>

[13] *What is Ryu Controller?* 2016. Sdxcentral [online]. sdxcenral. [cit. 2019 11-14] Dostupné z: <https://www.sdxcentral.com/networking/sdn/definitions/what-is-ryu-controller/>

[14] *RYU SDN Framework* [online]. RYU project team, c2014b [cit. 2019 11-21].Dostupné z: <https://osrg.github.io/ryu-book/en/html/index.html#>

[15] *Ryu Documentation* [online]. 2016. 4.30. ryu development team. [cit 2019 11-21] Dostupné z: <https://buildmedia.readthedocs.org/media/pdf/ryu/latest/ryu.pdf>

[16] SCOTT, Leo. 2016. *Ryu for Rapid Prototyping*. Inside-openflow [online]. [cit 2019 11-21] Dostupné z: <https://inside-openflow.com/2016/06/10/ryu-for-rapid-prototyping/>

[17] *A comparison between several Software Defined Networking controllers*. 2015. IEE [online]. Nis, Serbia: IEEE. [cit. 2019 11-17] Dostupné z: <https://ieeexplore.ieee.org/abstract/document/7357774>

[18] RAO, Sridhar. 2014. *SDN Series Part Three: NOX, the Original OpenFlow Controller*. Thenewstack [online]. [cit. 2019 11-19] Dostupné z: <https://thenewstack.io/sdn-series-part-iii-nox-the-original-openflow-controller/>

[19] HU, Tao, Zehua GUO, Peng YI, Thar BAKER and Julong LAN. 2018. *Multi-controller Based Software-Defined Networking: A Survey*. IEEE Access [online]. 6(1), 1-21. [cit. 2019 11-15] Dostupné z: <https://ieeexplore.ieee.org/document/8314783>

[20] SCOTT-HAYWARD, Sandra, Gemma O'CALLAGHAN and Sakir SEZER. 2013. *Sdn Security: A Survey*. In: 2013 IEEE SDN for Future Networks and

Services (SDN4FNS) [online]. Trento, Italy: IEEE, p. 1-7. [cit. 2019 10-16] Dostupné z: <http://ieeexplore.ieee.org/document/6702553/>

[21] SCOTT-HAYWARD, Sandra, Sriram NATARAJAN and Sakir SEZER. 2016. A *Survey of Security in Software Defined Networks*. IEEE Communications Surveys & Tutorials [online] [cit. 2019-10-15]. 18(1), 623-654. [cit. 2019 10-16] Dostupné z: <http://ieeexplore.ieee.org/document/7150550/

[22] ROUSE, Margaret. *Softwarově definované sítě (SDN)*. TechTarget Network [online][cit. 2019-10-16]. Dostupné z: <https://searchnetworking.techtarget.com/definition/software-defined-networking-SDN>

[23] *Performance Analysis of Software-Defined Networking (SDN)*. 2013. In: GELBERGER, Alexander, Niv YEMINI and Ran GILADI. 2013 IEEE 21st International Symposium on Modelling, Analysis and Simulation of Computer and Telecommunication Systems [online]. San Francisco, CA, USA: IEEE, p. 389-393. [cit. 2019-10-16] Dostupné z: <https://ieeexplore.ieee.org/abstract/document/6730793>

[24] GORANSSON, Paul and Chuck BLACK. [2014]. *Software defined networks: a comprehensive approach*. 2. Boston: Elsevier, Morgan Kaufmann.

[25] Introduction to OpenFlow. In: Youtube [online]. 7.10.2013 [cit. 2019-11-17]. Dostupné z: <https://youtu.be/l25Ukkmk6Sk>. Kanál uživatele David Mahler.

[26] GORANSSON, Paul, *Chuck BLACK and Timothy CULVER*. [2017]. Software defined networks: a comprehensive approach. Second edition. Singapore: Morgan Kaufmann.

[27] *OpenFlow Switch Specification Version 1.5.1. Open Networking Foundation* [online]. 2015 [cit. 2019-10-20]. Dostupné z: <https://www.opennetworking.org/wp-content/uploads/2014/10/openflow-switch-v1.5.1.pdf>

[28] MORAVCOVÁ, Klára. 2018. *Tvorba virtuálních síťových topologií pomocí softwarově definovaných sítí*. [cit. 2019 11-12] Brno. Diplomová práce. Vysoké Učení Technické. Vedoucí práce Doc. Ing. Vít Novotný, Ph.D.

[29] OpenFlow. 2019. *Programmability Configuration Guide*, Cisco IOS XE Gibraltar 16.11.x. 170 West Tasman Drive San Jose, CA 95134-1706 USA: Cisco Systems, 205 - 218. [cit. 2019 11-10]

[30] CHING-HAO, Chang; LIN, Ying-Dar. *OpenFlow Version Roadmap*. [online] 2015. [cit. 2019 11-13] Dostupné z: <http://speed.cis.nctu.edu.tw/~ydlin/miscpub/indep_frank.pdf>

[31] DONATO, Rick. 2017. *What is OpenFlow? Fir3net.* [online]. [cit. 2019 11-09] Dostupné z: <https://www.fir3net.com/Networking/Protocols/what-is-openflow.html>

[32] ENNS, R., et al. 2011. *Network Configuration Protocol (NETCONF): RFC 6241.* IETF, June, 2011. [online] [cit. 2019 11-25] Dostupné z: <https://tools.ietf.org/pdf/rfc6241.pdf>

[33] MOZUMDAR, Suvendu. 2018. *NETCONF and YANG: De facto Network Management for SDN.* Ixia [online]. [cit. 2019 11-25] Dostupné z: <https://www.ixiacom.com/company/blog/netconf-and-yang-de-facto-network-management-sdn>

[34] KREJČÍ, Radek. 2014. *NETCONF a YANG: NETCONF.* Nic [online]. [cit. 2019 11-25] Praha. Dostupné z: <https://www.nic.cz/public_media/IT14.2/prezentace/Radek_Krejci.pdf>

[35] BJORKLUND, M. 2010. *YANG - A Data Modeling Language for the Network Configuration Protocol (NETCONF)* [online]. [cit. 2019-11-26] , 1-173. Dostupné z: <https://tools.ietf.org/pdf/rfc6020.pdf>

[36] ROUSE, Margaret. 2017. *Network functions virtualization (NFV).* Searchnetworking. [online]. [cit. 2019 11-26] Dostupné z: <https://searchnetworking.techtarget.com/definition/network-functions-virtualization-NFV>

[37] TITTEL, Ed. *SDN vs. NFV: What's the difference?* Cisco [online]. [cit. 2019 11-26] Dostupné z: <https://www.cisco.com/c/en/us/solutions/software-defined-networking/sdn-vs-nfv.html>

[38] SCOTT, Leo. 2016. *Understanding the Ryu API: Dissecting Simple Switch.* Inside OpenFlow [online]. [cit. 2020-05-21] Dostupné z: <https://inside-openflow.com/2016/07/21/ryu-api-dissecting-simple-switch/>

[39] *Using VMware Workstation Pro.* 2019. Wmware [online]. [cit. 2020-05-21] Palo Alto Hillview Ave 3401. Dostupné z: <https://docs.vmware.com/en/VMware-Workstation-Pro/15.0/com.vmware.ws.using.doc/GUID-0EE752F8-C159-487A-9159-FE1F646EE4CA.html>

[40] *What Is Open vSwitch?* 2020. OvS OpenvSwitch [online]. [cit. 2020-05-21] Dostupné z: <http://docs.openvswitch.org/en/latest/intro/what-is-ovs/>

[41] SHARPE, Richard, WARNICKE, Ed (ed.). *Wireshark User's Guide*. Wireshark [online]. [cit. 2020-05-21] Dostupné z: <https://www.wireshark.org/docs/wsug_html_chunked/>

[42] LANTZ, Bob, Nikhil HANDIGOL, Brandon HELLER and Vimal JEYAKUMAR. 2018. *Introduction to Mininet*. Github [online]. [cit. 2020-05-20] Dostupné z: <https://github.com/mininet/mininet/wiki/Introduction-to-Mininet>

[43] TEAM, Mininet. 2018. *Mininet Walkthrough*. Mininet [online]. [cit. 2020-05-20] Dostupné z: <http://mininet.org/walkthrough/>

Seznam symbolů, veličin a zkratek

ACK	Acknowledgement
ACL	Access Control List
API	Application Programming Interface
ASIC	Application Specific Integrated Circuit
BGP	Border Gateway Protocol
BGP-LS	Border Gateway Protocol - Link State
DHCP	Dynamic Host Configuration Protocol
DNS	Domain Name System
DoS	Denial Of Service
DSCP	Differentiated Services Code Point
FIB	Forwarding Information Base
FTP	File Transfer Protocol
GUI	Graphical User Interface
HP	Hewlett Packard
HTTP	Hypertext Transfer Protocol
ICMP	Internet Control Message protocol
ID	Identification
IETF	Internet Engineering Task Force
IGP	Interior Gateway Protocol
IoT	Internet of Things
IP	Internet Protocol
IT	Information Technology
LACP	Link Aggregation Control Protocol
LC	Line Card
LISP	List Processing
LSA	Link State Advertisment
MAC	Media Access Control
MOD	Modification
NETCONF	Network Configuration Protocol
NFV	Network Functions Virtualization
OF-config	OpenFlow Configuration and Management Protocol
OFP	Open Fast Path
ONOS	Open Network Operating System
OSPF	Open Shortest Path First
OVS	Open vSwitch
OVSDB	Open vSwitch Database Management Protocol
OXM	OpenFlow Extensible Match

PC	Personal Computer
PCEP	Path Computation Element Communication Protocol
QoS	Quality of Services
REST	Representational state transfer
RIB	Routing Information Base
RIP	Routing Information Protocol
RP	Route Processor
RPC	Remote Procedure Call
SDN	Software Defined Network
SNMP	Simple Network Management Protocol
SSH	Secure Shell
SYN	Synchronization
TCAM	Ternary Content Addressable Memory
TCP	Transmission Control Protocol
TLS	Transport Layer Security
TLV	Type Lenght Value
URI	Uniform Resource Identifier
URL	Uniform Resource Locator
VM	Virtual Machine
VMRC	VMware Remote Console
VPN	Virtual Private Network
VRF	Virtual Rounting and Forwarding
vSwitch	Virtual Switch
WAN	Wide Area Network
XML	Extensible Markup Language
XPath	XML Path Language
YANG	Yet Another Next Generation

Seznam příloh

A Topologie

A.1 Zdrojový kód topologie použité pro firewall

Na výpisu níže naleznete zápis kódu pro topologii v programovacím jazyce python.

```
#!/usr/bin/python

"""Firewall topolgy

Four switches, where any of them have four hosts:


              Switch2
            /         \
      Switch1         Switch3


"""

from mininet.topo import Topo
from mininet.net import Mininet
from mininet.log import setLogLevel
from mininet.cli import CLI
from time import sleep
from mininet.node import OVSSwitch, Controller,
RemoteController



class SingleSwitchTopo(Topo):
    "Four switches, where any of them is connected to
     4 hosts."
    def build(self):
        s1 = self.addSwitch('s1')
        s2 = self.addSwitch('s2')
        s3 = self.addSwitch('s3')

        h1 = self.addHost('h1', mac="00:00:00:00:01:01",
```

```
        ip="192.168.1.1/24")
        h2 = self.addHost('h2', mac="00:00:00:00:01:02",
        ip="192.168.1.2/24")
        h3 = self.addHost('h3', mac="00:00:00:00:01:03",
        ip="192.168.1.3/24")

        h4 = self.addHost('h4', mac="00:00:00:00:02:01",
        ip="192.168.1.4/24")
        h5 = self.addHost('h5', mac="00:00:00:00:02:02",
        ip="192.168.1.5/24")
        h6 = self.addHost('h6', mac="00:00:00:00:02:03",
        ip="192.168.1.6/24")

        h7 = self.addHost('h7', mac="00:00:00:00:03:01",
        ip="192.168.1.7/24")
        h8 = self.addHost('h8', mac="00:00:00:00:03:02",
        ip="192.168.1.8/24")
        h9 = self.addHost('h9', mac="00:00:00:00:03:03",
        ip="192.168.1.9/24")

        self.addLink(s1, s2)
        self.addLink(s2, s3)

        self.addLink(s1, h1, 3, 0)
        self.addLink(s1, h2, 4, 0)
        self.addLink(s1, h3, 5, 0)

        self.addLink(s2, h4, 3, 0)
        self.addLink(s2, h5, 4, 0)
        self.addLink(s2, h6, 5, 0)

        self.addLink(s3, h7, 3, 0)
        self.addLink(s3, h8, 4, 0)
        self.addLink(s3, h9, 5, 0)

if __name__ == '__main__':
    setLogLevel('info')
    topo = SingleSwitchTopo()
    c1 = RemoteController('c1', ip='192.168.3.5:6653')
```

```
    net = Mininet(topo=topo, controller=c1)
    net.start()
    sleep(1)
    CLI(net)
    if net.stop():
       net.clean()
```

A.2 Zdrojový kód topologie použité pro směrování

Na výpisu níže naleznete zápis kódu pro topologii v programovacím jazyce python.

```
#!/usr/bin/python

"""Firewall topolgy

Four switches, where any of them have three hosts:

              Switch4
                 |
              Switch2
            /         \
      Switch1         Switch3


"""

from mininet.topo import Topo
from mininet.net import Mininet
from mininet.log import setLogLevel, info
from mininet.node import Node
from mininet.cli import CLI
from time import sleep
from mininet.node import OVSSwitch, Controller,
RemoteController

class TopoWithThreeSwitch(Topo):
    "Four switches, where any of them
    is connected to 3 hosts."
```

```
def build(self):

    s1 = self.addSwitch('s1', ip="192.168.1.1")
    s2 = self.addSwitch('s2', ip="192.168.2.1")
    s3 = self.addSwitch('s3', ip="192.168.3.1")
    s4 = self.addSwitch('s4', ip="192.168.4.1")

    h1 = self.addHost('h1', mac="00:00:00:00:01:01",
    ip="192.168.1.2/24",
       defaultRoute='via 192.168.1.1')
    h2 = self.addHost('h2', mac="00:00:00:00:01:02",
    ip="192.168.1.3/24",
       defaultRoute='via 192.168.1.1')
    h3 = self.addHost('h3', mac="00:00:00:00:01:03",
    ip="192.168.1.4/24",
       defaultRoute='via 192.168.1.1')

    h4 = self.addHost('h4', mac="00:00:00:00:02:01",
    ip="192.168.2.2/24",
       defaultRoute='via 192.168.2.1')
    h5 = self.addHost('h5', mac="00:00:00:00:02:02",
    ip="192.168.2.3/24",
       defaultRoute='via 192.168.2.1')
    h6 = self.addHost('h6', mac="00:00:00:00:02:03",
    ip="192.168.2.4/24",
       defaultRoute='via 192.168.2.1')

    h7 = self.addHost('h7', mac="00:00:00:00:03:01",
    ip="192.168.3.2/24",
       defaultRoute='via 192.168.3.1')
    h8 = self.addHost('h8', mac="00:00:00:00:03:02",
    ip="192.168.3.3/24",
       defaultRoute='via 192.168.3.1')
    h9 = self.addHost('h9', mac="00:00:00:00:03:03",
    ip="192.168.3.4/24",
       defaultRoute='via 192.168.3.1')

    h10 = self.addHost('h10', mac="00:00:00:00:04:01",
    ip="192.168.3.2/24",
```

```
            defaultRoute='via 192.168.4.1')
        h11 = self.addHost('h11', mac="00:00:00:00:04:02",
        ip="192.168.3.3/24",
            defaultRoute='via 192.168.4.1')
        h12 = self.addHost('h12', mac="00:00:00:00:04:03",
        ip="192.168.3.4/24",
            defaultRoute='via 192.168.4.1')

        self.addLink(s1, s2, 1, 1)
        self.addLink(s2, s3, 2, 1)
        self.addLink(s2, s4, 3, 1)

        self.addLink(s1, h1, 3, 1)
        self.addLink(s1, h2, 4, 1)
        self.addLink(s1, h3, 5, 1)

        self.addLink(s2, h4, 4, 1)
        self.addLink(s2, h5, 5, 1)
        self.addLink(s2, h6, 6, 1)

        self.addLink(s3, h7, 3, 1)
        self.addLink(s3, h8, 4, 1)
        self.addLink(s3, h9, 5, 1)

        self.addLink(s4, h10, 3, 1)
        self.addLink(s4, h11, 4, 1)
        self.addLink(s4, h12, 5, 1)

if __name__ == '__main__':
    setLogLevel('info')
    topo = TopoWithThreeSwitch()
    c1 = RemoteController('c1', ip='192.168.3.5:6653')
    net = Mininet(topo=topo, controller=c1)
    net.start()
    sleep(1)
    CLI(net)
    if net.stop():
       net.clean()
```

A.3 Zdrojový kód topologie přepínač se 4 hosty

Na výpisu níže naleznete zápis kódu pro topologii v programovacím jazyce python.

```
#!/usr/bin/python

"""Simulation topolgy

One switch and 4 hosts:

           host
            |
            |
host --- switch --- host
            |
            |
           host
"""

from mininet.topo import Topo
from mininet.net import Mininet
from mininet.log import setLogLevel
from mininet.cli import CLI
from time import sleep
from mininet.node import OVSSwitch, Controller,
RemoteController



class SingleSwitchTopo(Topo):
    "Single switch connected to 4 hosts."
    def build(self):
        s1 = self.addSwitch('s1')
        h1 = self.addHost('h1', mac="00:00:00:00:10:11",
        ip="192.168.1.1/24")
        h2 = self.addHost('h2', mac="00:00:00:00:10:12",
        ip="192.168.1.2/24")
        h3 = self.addHost('h3', mac="00:00:00:00:10:13",
        ip="192.168.1.3/24")
```

```
        h4 = self.addHost('h4', mac="00:00:00:00:10:14",
        ip="192.168.1.4/24")

        self.addLink(h1, s1, 0, 1)
        self.addLink(h2, s1, 0, 2)
        self.addLink(h3, s1, 0, 3)
        self.addLink(h4, s1, 0, 4)

if __name__ == '__main__':
    setLogLevel('info')
    topo = SingleSwitchTopo()
    c1 = RemoteController('c1', ip='192.168.3.5:6653')
    switch=OVSSwitch
    net = Mininet(topo=topo, controller=c1)
    net.start()
    sleep(1)
    CLI(net)
    if net.stop():
       net.clean()
```

A.4 YANG model zapsaný ve formátu JSON

Na výpisu níže je možné vidět YANG model, sloužící k nastavení počáteční konfigurace datových toků.

```
    {
  "format": "FLOWMANAGER_v1.0",
  "switches": [
    1
  ],
  "meters": [
    {
      "1": []
    }
  ],
  "groups": [
    {
      "1": []
    }
```

```
],
"flows": [
  {
    "1": [
      {
        "priority": 2500,
        "cookie": 0,
        "idle_timeout": 0,
        "hard_timeout": 0,
        "byte_count": 3836,
        "duration_sec": 291,
        "duration_nsec": 232000000,
        "packet_count": 46,
        "length": 64,
        "flags": 0,
        "actions": [
          "GOTO_TABLE:1"
        ],
        "match": {},
        "table_id": 0
      },
      {
        "priority": 1,
        "cookie": 0,
        "idle_timeout": 0,
        "hard_timeout": 0,
        "byte_count": 2940,
        "duration_sec": 3119,
        "duration_nsec": 717000000,
        "packet_count": 30,
        "length": 80,
        "flags": 0,
        "actions": [],
        "match": {
          "eth_type": 2048,
          "ipv4_src": "192.168.1.1",
          "ipv4_dst": "192.168.1.3"
        },
        "table_id": 1
```

```
},
{
  "priority": 1,
  "cookie": 0,
  "idle_timeout": 0,
  "hard_timeout": 0,
  "byte_count": 196,
  "duration_sec": 3010,
  "duration_nsec": 477000000,
  "packet_count": 2,
  "length": 80,
  "flags": 0,
  "actions": [],
  "match": {
    "eth_type": 2048,
    "ipv4_src": "192.168.1.2",
    "ipv4_dst": "192.168.1.4"
  },
  "table_id": 1
},
{
  "priority": 1,
  "cookie": 0,
  "idle_timeout": 0,
  "hard_timeout": 0,
  "byte_count": 294,
  "duration_sec": 2999,
  "duration_nsec": 492000000,
  "packet_count": 3,
  "length": 80,
  "flags": 0,
  "actions": [],
  "match": {
    "eth_type": 2048,
    "ipv4_src": "192.168.1.3",
    "ipv4_dst": "192.168.1.2"
  },
  "table_id": 1
},
```

```
{
  "priority": 0,
  "cookie": 0,
  "idle_timeout": 0,
  "hard_timeout": 0,
  "byte_count": 6076,
  "duration_sec": 2947,
  "duration_nsec": 571000000,
  "packet_count": 98,
  "length": 64,
  "flags": 0,
  "actions": [
    "GOTO_TABLE:2"
  ],
  "match": {},
  "table_id": 1
},
{
  "priority": 1,
  "cookie": 0,
  "idle_timeout": 0,
  "hard_timeout": 0,
  "byte_count": 196,
  "duration_sec": 2409,
  "duration_nsec": 117000000,
  "packet_count": 2,
  "length": 104,
  "flags": 0,
  "actions": [
    "OUTPUT:2"
  ],
  "match": {
    "eth_type": 2048,
    "ipv4_src": "192.168.1.1",
    "ipv4_dst": "192.168.1.2"
  },
  "table_id": 2
},
{
```

```
    "priority": 1,
    "cookie": 0,
    "idle_timeout": 0,
    "hard_timeout": 0,
    "byte_count": 196,
    "duration_sec": 2325,
    "duration_nsec": 900000000,
    "packet_count": 2,
    "length": 104,
    "flags": 0,
    "actions": [
      "OUTPUT:1"
    ],
    "match": {
      "eth_type": 2048,
      "ipv4_src": "192.168.1.2",
      "ipv4_dst": "192.168.1.1"
    },
    "table_id": 2
  },
  {
    "priority": 1,
    "cookie": 0,
    "idle_timeout": 0,
    "hard_timeout": 0,
    "byte_count": 0,
    "duration_sec": 2260,
    "duration_nsec": 766000000,
    "packet_count": 0,
    "length": 104,
    "flags": 0,
    "actions": [
      "OUTPUT:1"
    ],
    "match": {
      "eth_type": 2048,
      "ipv4_src": "192.168.1.4",
      "ipv4_dst": "192.168.1.1"
    },
```

```
  "table_id": 2
},
{
  "priority": 1,
  "cookie": 0,
  "idle_timeout": 0,
  "hard_timeout": 0,
  "byte_count": 0,
  "duration_sec": 2198,
  "duration_nsec": 11000000,
  "packet_count": 0,
  "length": 104,
  "flags": 0,
  "actions": [
    "OUTPUT:4"
  ],
  "match": {
    "eth_type": 2048,
    "ipv4_src": "192.168.1.1",
    "ipv4_dst": "192.168.1.4"
  },
  "table_id": 2
},
{
  "priority": 1,
  "cookie": 0,
  "idle_timeout": 0,
  "hard_timeout": 0,
  "byte_count": 0,
  "duration_sec": 2143,
  "duration_nsec": 777000000,
  "packet_count": 0,
  "length": 104,
  "flags": 0,
  "actions": [
    "OUTPUT:3"
  ],
  "match": {
    "eth_type": 2048,
```

```
    "ipv4_src": "192.168.1.4",
    "ipv4_dst": "192.168.1.3"
  },
  "table_id": 2
},
{
  "priority": 1,
  "cookie": 0,
  "idle_timeout": 0,
  "hard_timeout": 0,
  "byte_count": 0,
  "duration_sec": 2121,
  "duration_nsec": 413000000,
  "packet_count": 0,
  "length": 104,
  "flags": 0,
  "actions": [
    "OUTPUT:4"
  ],
  "match": {
    "eth_type": 2048,
    "ipv4_src": "192.168.1.3",
    "ipv4_dst": "192.168.1.4"
  },
  "table_id": 2
},
{
  "priority": 0,
  "cookie": 0,
  "idle_timeout": 0,
  "hard_timeout": 0,
  "byte_count": 84,
  "duration_sec": 169,
  "duration_nsec": 258000000,
  "packet_count": 2,
  "length": 80,
  "flags": 0,
  "actions": [
    "OUTPUT:CONTROLLER"
```

```
          ],
          "match": {},
          "table_id": 2
        }
      ]
    }
  ]
}
```

B Laboratorní návod

B.1 Cíl

Hlavním cílem této laboratorní úlohy je osvojit si základy Softwarově definovaných sítí a používání kontroléru Ryu. Dále pak naučit se pracovat s datovými toky a rozpoznat síťovou komunikaci pomocí protokolu OpenFlow.

B.2 Vybavení pracoviště

2xPC, VMware, Ryu, Wireshark, OpenVSwitch a Mininet.

B.3 Úkoly

- Osvojení základních poznatků o softwarově definovaných sítích a protokolu OpenFlow.
- Seznámení se základními příkazy Mininetu, OpenVSwitche a kontroléru Ryu.
- Seznámení se se základními moduly kontroléru Ryu, jmenovitě SimpleSwitch, Firewall a Router.

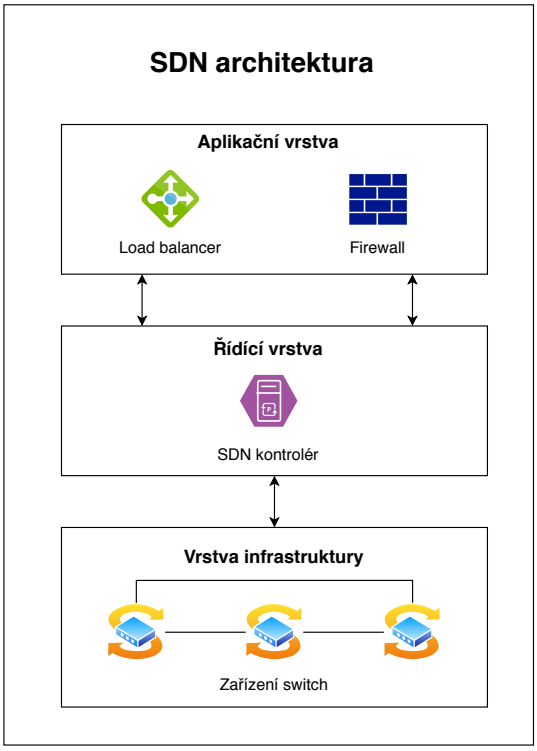
B.4 Teoretický úvod

B.4.1 Softwarově definované sítě

SDN je architektura, která poskytuje dynamičnost, snadnější správu a větší cenovou dostupnost dnešních sítí. Základní vlastností je oddělení řídící a datové roviny, čímž usnadňuje práci síťovým zařízením. Centrálním mozkem této sítě je kontrolér, který odesílá instrukce přepínačům a tím řídí celý její chod. Základním komunikačním protokolem je OpenFlow, umožňuje komunikaci s nižšími vrstvami architektury. [6]

Architektura SDN

Základem SDN je rozdělení zařízení na dvě části: řídící, která se stará o řízení provozu a správu datových toků a datovou, jenž má za úkol přesouvat pakety z portu na port. Architektura této technologie je rozdělena na tři vrstvy: aplikační, řídící a vrstvu infrastruktury. [6] Aplikační vrstva se nachází na nejvyšší vrstvě a poskytuje aplikace pro řídící vrstvu, které předávají kontroléru informace o chování sítě skrze API. V poslední řadě se stará o vytváření tzv. abstraktního náhledu na síťovou topologii, jenž je důležitý při rozhodování kontroléru. Řídící vrstva se stará řízení



Obr. B.1: Architektura SDN [22]

a správu sítě. Nachází se zde kontrolér, který pracuje jako logická jednotka, jenž dostává instrukce nebo požadavky od aplikací na aplikační vrstvě. Skládá se ze severního a jižního rozhraní. Severní zprostředkovává komunikace s aplikační vrstvou prostřednictví REST API a jižní se využívá ke komunikaci s vrstvou infrastruktury, nachází se zde protokoly Openflow a NETCONF. Poslední vrstva je infrastrukturní. V této části se nachází síťová zařízení. Jedná se o fyzickou vrstvu této architektury. [7] [8] [9]

B.4.2 Protokol Openflow

OpenFlow je jedním z důležitých neproprietárních protokolů SDN. Jedná se o nástroj, který definuje komunikaci mezi kontrolérem a přepínačem. Komunikace jeumožněna skrze zabezpečený kanál. K zabezpečení tohoto kanálu je ve většině případů využito TLS (Transport layer security) asymetrické kryptografie. K této komunikaci využívá sadu oboustranně odlišných zpráv, které si mezi sebou vzájemně posílají. Tyto zprávy umožňují kontroléru naprogramovat přepínač, tak aby byla umožněna co nejpřesnější kontrola a případná úprava síťového toku. [6]

Openflow zprávy

Mezi základní zprávy patří: Hello, Packet-in, Port-status, Packet-out, Features, Flow-mod.

- **Hello:** Tuto zprávu si vyměňuje kontrolér s přepínačem při zahájení spojení.
- **Packet-in:** Touto zprávou dostává kontrolér informaci od přepínače o nenalezení paketu s podobným chováním v jeho tabulce toků a žádá o pomoc. Odpovědí na tuto zprávu je Packet-out.
- **Echo:** Zpráva sloužící k ověřování komunikace mezi kontrolérem a přepínačem.
- **Port-status:** Slouží jako informační zpráva v případě, že dojde ke změně stavu portu přepínače. Je odeslána například v případě, že port je v základu nastaven jako aktivní a dojde k jeho manuálnímu vypnutí, nebo k výpadku linky.
- **Packet-out:** Jedná se zprávu, která obsahuje instrukce k zpracování specifického paketu. Tyto zprávy se využívají v případě, že je třeba odeslat zprávu přímo na specifický port přepínače, nebo slouží jako odpověď na zprávu Packet-in.
- **Features:** Kontrolér se dotazuje zprávou features request na identitu a základní vlastnosti přepínače. Ten zpětně odešle zprávu features reply, ve které se budou nacházet vyžádané informace. Této zprávy se využívá při vytváření kanálu OpenFlow.
- **Flow-mod:** Tato zpráva slouží k odeslání informací o toku přepínači, který ji použije jako předlohu k vytvoření datového toku v tabulce toků. [6] [27] [25]

B.4.3 Openflow přepínač

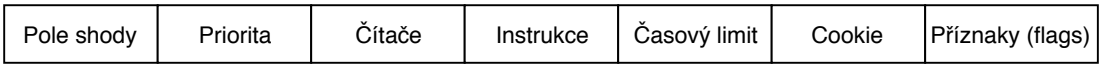
Jedná se o přepínač, který disponuje protokolem Openflow a podporuje některý z kontrolérů. Jeho vnitřní struktura je složená z tabulek toků, skupin a měření. Tyto tabulky jsou využity v logice přepínače. Základním principem je porovnávání příchozích datových toků (proudů paketů) s toky uloženými v tabulce. Paket, který projde kontrolou může být buď odeslán na daný výstupní port, to nastává pokud je v tabulce nalezena shoda, nebo odeslán kontroléru ke zpracování a vyhodnocení, anebo je zahozen. Celá tato komunikace probíhá po zabezpečeném kanálu pomocí zpráv Packet-IN, Packet-OUT a Flow-Mod.

Tabulka toků

Jak bylo zmíněno výše, přepínač k rozhodování využívá tzv. tabulku toků, jejíž obsah je zobrazen na B.2. Pokud se nachází více tabulek za sebou, řadí se do tzv. pipeline,

kde jsou vybírány podle priority. Jednotlivé toky, které jsou umístěny v tabulce se skládají z těchto částí:

- **Pole shody:** Nachází se zde pravidla shody vstupního toku. Obsahuje vstupní port, záhlaví paketů, a metadata zadaná v předchozí tabulce.
- **Priorita:** Podle této položky je rozhodnuto pořadí toků.
- **Čítače:** Počet paketů, které odpovídají shodě v tabulce.
- **Instrukce (akce):** Udává, co se má s taným tokem stát.
- **Časový limit:** Maximální doba nečinnosti daného vstupu toku.
- **Cookie:** Identifikátor vstupu toku.
- **Příznaky (flags):** Další doplňující informace. [24] [27]



Obr. B.2: Informace datového toku [27]

B.4.4 OpenVSwitch

Jedná se o vícevrstvý softwarový přepínač licencovaný pod licencí Open Source Apache 2. Využívá se nejčastěji jako virtuální přepínač, který je využívá v prostředí, kde se nachází virtuální stroje. Podporuje většinu známých virtualizačních prostředí založených na Linuxu, například KVM, VMware a VirtualBox. Mezi hlavní části tohoto přepínače patří: [40]

- **ovs-vswitchd:** Démon, který implementuje přepínač spolu s doprovodným linuxovým modulem jádra pro přepínání podle toků.
- **ovsdb-server:** Odlehčený databázový server, jenž poskytuje konfiguraci démonovi.
- **ovs-dpctl:** Nástroj pro konfiguraci jádra přepínače.
- **ovs-vsctl:** Služba pro dotazování a aktualizaci konfigurace démona.
- **ovs-appctl:** Nástroj, kterým jsou odesílány příkazy běžícímu OVS démonovi. [40]

B.4.5 Ryu

V této laboratorní úloze bude využit open source kontrolér Ryu. Je napsaný v jazyce Python a volně dostupný na GitHubu. Jeho výhodou je volná úprava zdrojového kódu. Je tvořen sadou modulů, které lze mezi sebou kombinovat. V této úloze bude využito modulů SimpleSwitch, který slouží k obsluze L2 přepínačů, Firewall, jímž je

možné nastavovat jednotlivá pravidla firewallu pro všechny přepínače v síti a Router, jenž se využívá ke konfiguraci statického směrování L3 přepínačů. Postupně se s těmito moduly seznámíte v jednotlivých úkolech. Ten kontrolér podporuje protokoly Openflow ve vezích 1.0 - 1.5, NETCONF a Of-config. Dále disponuje REST API, skrze které je mu umožněn přístup k přepínačům a jejich konfigurace (některé moduly ho využívají, například Router a Firewall). Tento kontrolér má dostupné grafické webové rozhraní, kde je možno prohlédnout si topologii a sledovat síťový provoz. Dále jsou zde uloženy informace o konfiguraci zařízení, například směrovací tabulka, pravidla firewallu a také záznamy o jednotlivých datových tocích. [13]

B.4.6 Mininet

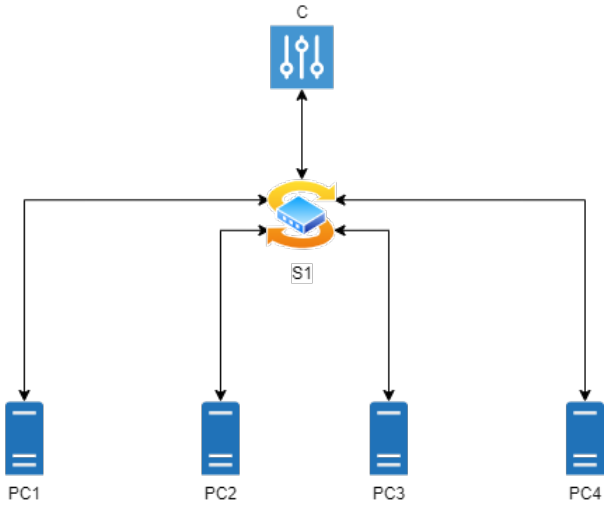
Mininet poskytuje virtuální vývojové prostředí pro vytváření a testování síťových topologií. Podporuje velkou řadu síťových protokolů a co je pro nás důležité, umožňuje vytvářet SDN topologie. Topologie vytvořené v tomto prostředí je možné přenášet na různé počítače se systémem Linux a propojovat je s fyzickými topologiemi. Pro vytvoření takové topologie se využívá programovací jazyk Python. Zde je přehled některých častých parametrů: [42]

- **Topo:** Základní třída topologie v Mininetu.
- **build():** Metoda na sestavení topologie.
- **addSwitch:** Přidá přepínač do topologie.
- **addHost:** Přidá hosta (PC) do topologie.
- **addLink:** Přidá obousměrný odkaz do topologie (slouží k propojení přepínačů a hostů mezi sebou).
- **Mininet:** Hlavní třída k vytváření a správě sítě.
- **start():** Spustí topologii.
- **stop():** Zastaví topologii. [43]

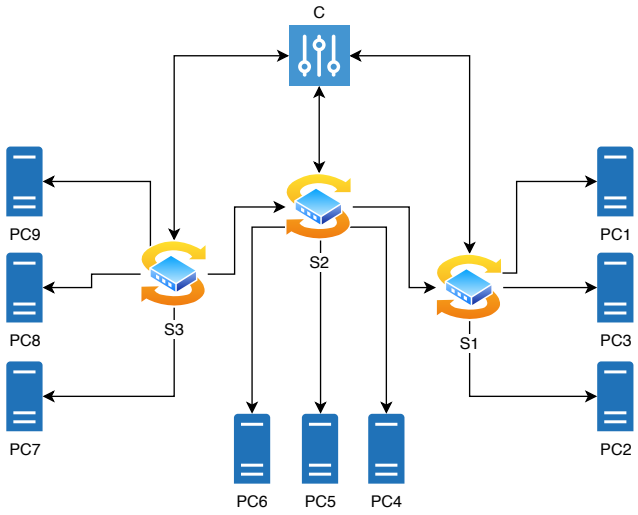
B.5 Seznámení s pracovištěm

Úloha vás má seznámit s technologií SDN. K její realizaci budete využívat dva virtuální stroje. Jedním z nich je Kontrolér, na kterém se nachází kontrolér Ryu a flow manager. Druhý virtuální stroj je Mininet. Na tomto stroji se nachází software Mininet a OpenVSwitch. Taktéž tu naleznete zdrojové kódy topologií, které budou popsány dále. Následující topologie budou použity pro realizaci laboratorní úlohy. První z nich bude použita při seznamování se s technologií SDN. Skládá se z jednoho přepínače S1 a 4 PC. Druhá v topologie je určena pro konfiguraci pravidel firewallu. Skládá se ze 3 přepínačů a každý z nich disponuje 3 PC. Poslední topologie slouží k realizaci směrování mezi L3 přepínači. Skládá se ze 4 přepínačů a každému z nich

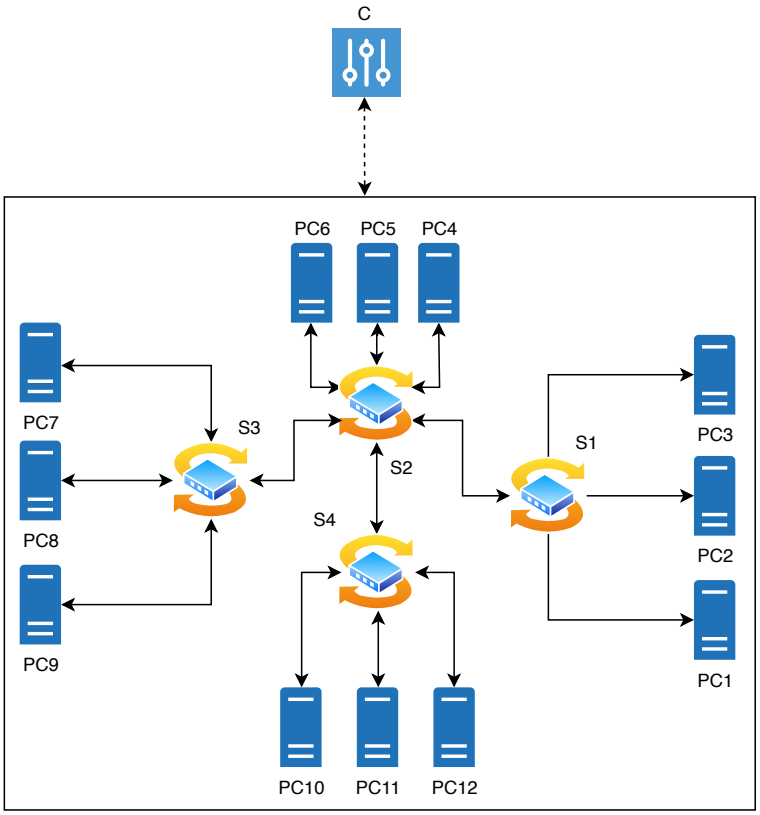
jsou přiděleny 3 PC. Přepínač S2 slouží jako centrální uzel topologie a proto zde bude směrování nejnáročnější. Zdrojové kódy jednotlivých topologií si můžete prohlédnout v záložce scripts na VM Mininet.



Obr. B.3: Topologie použita pro seznámení s SDN



Obr. B.4: Topologie pro realizaci firewallu



Obr. B.5: Topologie pro směrování

B.6 Postup řešení

B.7 První část laboratorní úlohy

B.7.1 Úkol 1: Seznámení s technologií softwarově definovaných sítí

Cílem tohoto úkolu je seznámit se základními nástroji kontroléru Ryu, OVS a Mininet. Dále pak zachytit OpenFlow zprávy, seznámit se základními položkami datových toků a upravit konfiguraci pomocí Flow manageru.

1. Přes VMRC se připojte na virtuální stroje Kontroler (PC1) a Mininet(PC2). Zadejte přihlašovací údaje student/student.
2. Přepněte se do virtuálního stroje (VM) Kontoler a seznamte se prostředím Ubuntu.
3. Zapněte konzolové okno (černý čtvereček v liště nalevo).
4. Nachází se zde kontrolér. Ten lze spustit příkazem:

   ```
   sudo ryu-manager <cesta k modulu>
   ```

5. Zapněte modul simpleswitch příkazem:

   ```
   sudo ryu-manager ryu/ryu/app/MatchByIP
   ```

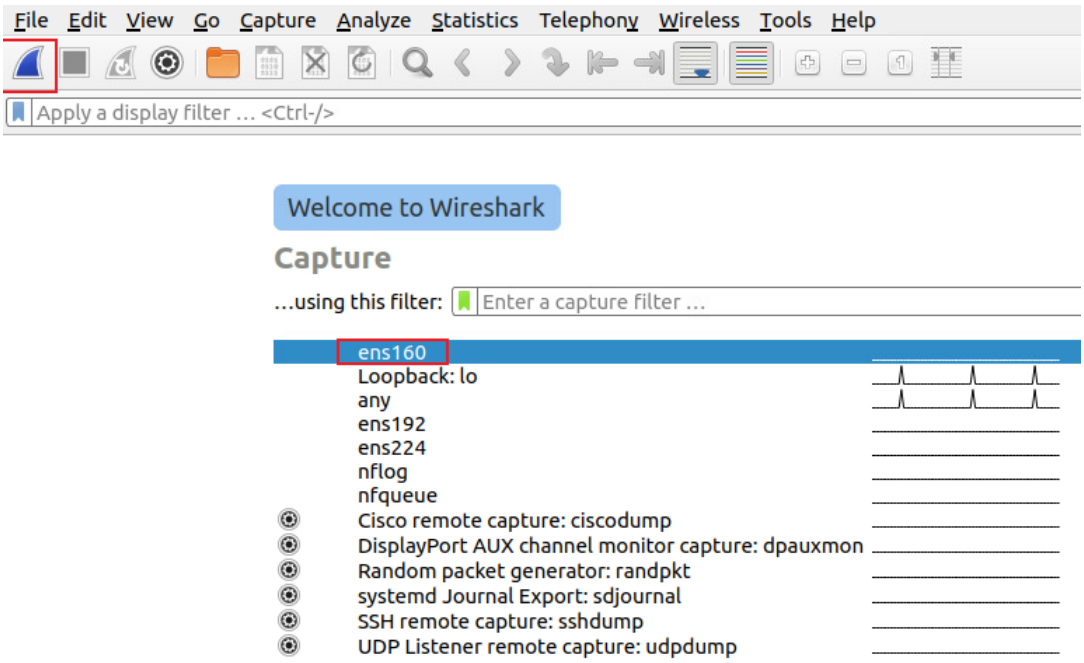
   Jedná se o modul, který nastaví kontrolér tak, aby byl schopen řídit přepínače.
6. Přepněte se na (VM) Mininet.
7. V adresáři scripts se nachází skripty potřebné k realizaci této úlohy. Do adresáře se přepnete příkazem:

   ```
   cd scripts
   ```

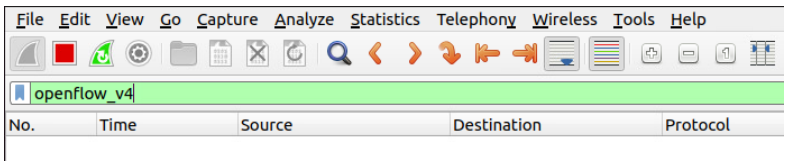
8. Spusťte aplikace pro zachytávání síťového provozu Wireshark (modrá ploutvička). Klikněte na rozhraní ens160 a spusťte zachytávání viz B.6. Nastavte filtr jak je zobrazeno na B.7.
9. Otevřete konzolové okno a vytvořte topologii příkazem:

   ```
   sudo ./topologyMininet.py
   ```

10. Po spuštění se vám zobrazí konzolové okno Mininetu. Zde se můžete příkazem links podívat na zapojení topologie.



Obr. B.6: Spuštění zachytávání provozu na ens160



Obr. B.7: Nastavení filtru pro OpenFlow

11. Když je topologie vytvořena, tak se automaticky propojí s kontrolérem a vymění si mezi sebou základní zprávy. Pokud jste postupovali správně, tak by jste měli vidět následující sled zpráv (zakladni zpravy).
12. Jednotlivé zprávy si projděte a pokuste se zjistit k čemu slouží (jako nápověda vám poslouží teoretický úvod). Podrobné informace o zprávě, kterou máte označenou jsou uloženy ve výpisu pod zprávami. Nejdůležitější informace naleznete po rozbalení záložky OpenFlow 1.3.
13. Přepněte se zpět na kontrolér.
14. Všimněte si, že kontrolér přijímá zprávy packet-in od přepínače. Těmito zprávami si předávají mezi sebou informace.
15. Nyní si otestujete jak kontrolér spolupracuje s přepínačem.
16. Přepněte se na Mininet a do konzole Mininetu napište následující příkaz:

```
h1 ping h2 -c 1
```

Tím se odešle ICMP zpráva z hostu h1 na h2)

17. Ve Wiresharku byla tato akce zaznamenána viz B.8.

openflow_v4

No.	Time	Source	Destination	Protocol	Length	Info
204	322.201543050	10.100.107.11	10.100.107.10	OpenFlow	150	Type: OFPT_PACKET_IN
205	322.203029473	10.100.107.10	10.100.107.11	OpenFlow	148	Type: OFPT_PACKET_OUT
207	322.203336930	10.100.107.11	10.100.107.10	OpenFlow	150	Type: OFPT_PACKET_IN
208	322.203980936	10.100.107.10	10.100.107.11	OpenFlow	148	Type: OFPT_PACKET_OUT
210	322.204213017	10.100.107.11	10.100.107.10	OpenFlow	206	Type: OFPT_PACKET_IN
211	322.205121301	10.100.107.10	10.100.107.11	OpenFlow	170	Type: OFPT_FLOW_MOD
213	322.205140547	10.100.107.10	10.100.107.11	OpenFlow	204	Type: OFPT_PACKET_OUT
215	322.205591738	10.100.107.11	10.100.107.10	OpenFlow	206	Type: OFPT_PACKET_IN
216	322.206358152	10.100.107.10	10.100.107.11	OpenFlow	170	Type: OFPT_FLOW_MOD
218	322.206377168	10.100.107.10	10.100.107.11	OpenFlow	204	Type: OFPT_PACKET_OUT
220	327.206665027	10.100.107.11	10.100.107.10	OpenFlow	74	Type: OFPT_ECHO_REQUEST
221	327.207189660	10.100.107.10	10.100.107.11	OpenFlow	74	Type: OFPT_ECHO_REPLY
223	327.392914724	10.100.107.11	10.100.107.10	OpenFlow	150	Type: OFPT_PACKET_IN
224	327.393782069	10.100.107.10	10.100.107.11	OpenFlow	148	Type: OFPT_PACKET_OUT
226	327.394014086	10.100.107.11	10.100.107.10	OpenFlow	150	Type: OFPT_PACKET_IN
227	327.394628961	10.100.107.10	10.100.107.11	OpenFlow	148	Type: OFPT_PACKET_OUT

Obr. B.8: Záznam ICMP zprávy

18. Jak můžete vidět tak bylo odesláno několik zpráv. Jak již bylo řečeno zprávami packet-in a packet-out si vyměnil přepínač a kontrolér informace. První zprávou říká kontroléru, že nemá pro daný paket žádný datový tok ve své tabulce (položka Reason).
19. Další zpráva slouží k odeslání instrukcí přepínači, v tomto případě je mu oznámeno, že má záplavově odeslat ARP request (položka action) a odpoví mu stanice, kterou hledá.
20. Kontrolér po obdržení této zprávy odešle zprávu FLOW MOD, kterou nastaví datový tok na přepínači. Zároveň kontrolér odesílá zprávu packet-out, čímž povoluje přepínači odeslat ICMP zprávu. Výše zmíněná situace se opakuje i pro druhou stanici (druhý FLOW MOD).
21. Přidané datové toky si můžete prohlédnout na VM Mininet spuštěním druhého konzolového okna a zadáním příkazu:

    ```
    sudo ovs-ofctl dump-flows s1
    ```

    a případně zadáním hesla student. Můžete zde vidět i první datový, který se přidá při propojení přepínače s kontrolérem. Je nastaven tak, aby odesílal veškerý neznámý provoz na kontrolér.
22. Ukončete modul kontroléru stisknutím kombinace kláves ctrl + c a zavřete topologii na VM Mininet příkazem "exit".
23. Poslední částí tohoto úkolu je nastavení datových toků pomocí Flow Manageru. Tato aplikace slouží k statickému nastavení datových toků. Vysvětlíme si zde vytvoření ACL pomocí těchto toků a pipelining tabulek toků.

24. Přepněte se na VM kontrolér a spusťte kontrolér spolu s Flow Managerem příkazem:

```
sudo ryu-manager --observe-links
ryu/ryu/app/MatchByIP.py
~/flowmanager/flowmanager.py
```

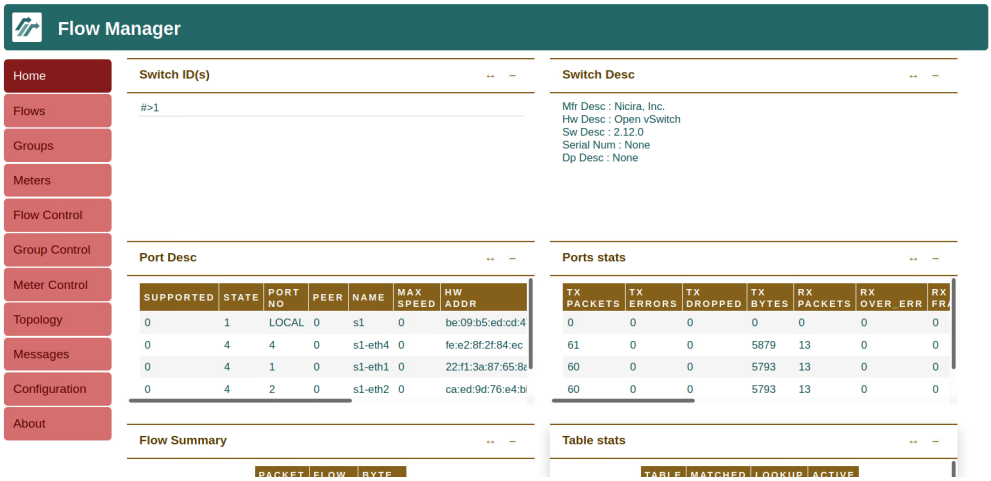
25. Dále na Mininetu vytvořte topologii příkazem:

```
sudo ./topologyMininet.py
```

26. Přepněte se zpět na kontrolér, otevřete prohlížeč a zadejte adresu:

```
localhost:8080/home/index.html
```

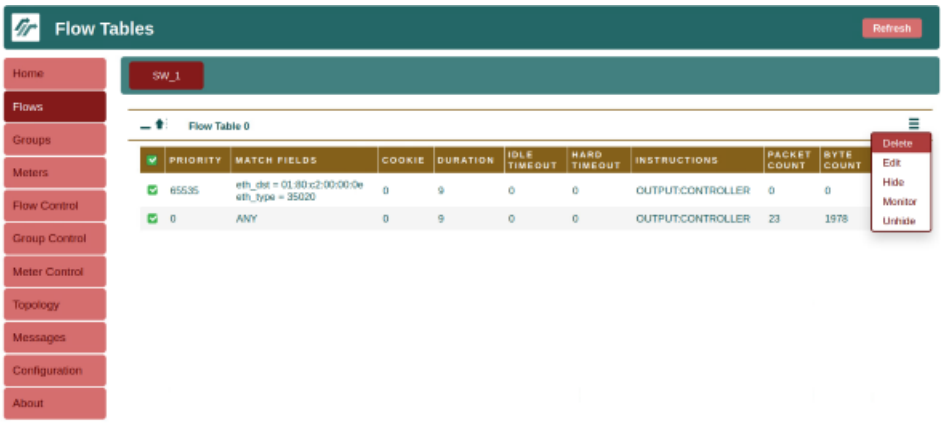
tím se dostanete do rozhraní Flow Manageru, které můžete vidět na B.9. Na



Obr. B.9: Rozhraní Flow Manageru

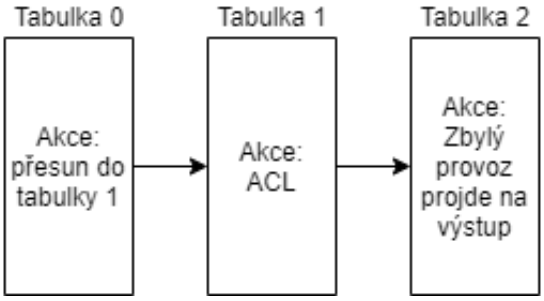
obrázku můžete vidět na levé straně záložky (popíšeme si jen ty co budou používány):Home: přehled všech informací, Flows: tabulka toků, Flow Control: úprava datových toků v tabulce, Topology: zobrazení topologie sítě a Configuration: zde lze uložit a nahrát konfiguraci ve formě YANG modelu jednotlivých toků a tabulek. Prohlédněte si jednotlivé záložky.

27. Po spuštění se přidaly toky kontroléru, tyto toky smažte pomocí následujícího obrázku. Na obrázku je zobrazena nabídka možností úpravy toků. Důležitá je volba edit, kterou lze vybraný datový tok editovat.



Obr. B.10: Smazání toků

28. Následně vložte novou konfiguraci. Přepněte se do záložky Configuration, klikněte na Choose Backup File, vyberte soubor "konfig.bk"a následně použijte tlačítko restore. Tímto se nahrála konfigurace potřebná k této části úlohy.
29. Konfigurace rozdělí datové toky do následujících tabulek, kde z tabulky 0 je všechen provoz přesunut do tabulky 1, kde jsou nastavena pravidla firewallu (zahodí provoz mezi H1 H3, H2 H4 a H3 H2, zbytek pošle ke kontrole do tabulky 2), tabulka 2 následně pošle zbývající provoz na výstup. Tomuto propojení se říká propojení do pipeline, využívá se toho při nastavování datových toků u větších topologií.



Obr. B.11: Tabulky toků

30. Prohlédněte si datové toky a pokuste se vytvořit tok, který bude ve druhé tabulce (Table 1) a bude zahazovat provoz mezi hostem H1 a H4. Jako inspiraci použijte toky s akcí DROP. Tyto toky jsou využity ke stejnému účelu, ale pro jiné hosty. Správnost můžete ověřit odesláním ICMP zprávy mezi H1 a H4 v Mininetu.
31. Výsledek vašeho úkolu ukažte vyučujícímu.
32. Vypněte modul na kontroléru (ctrl + c) a zavřete topologii (příkaz exit).

Otázky

1. K čemu slouží zpráva FLOW_MOD?
2. Jaké zprávy si mezi sebou vymění kontrolér a přepínač po propojení?
3. Jaké jsou podle vás výhody Softwarově definovaných sítí?
4. Jaký kontrolér je v úloze použit?
5. K čemu slouží flow manager?
6. Jaké jsou jeho výhody?

B.8 2. část laboratorní úlohy

B.8.1 Úkol 2: Nastavení pravidel firewallu prostřednictvím Ryu

Příprava prostředí

V této části si připravíte prostředí pro následnou konfiguraci firewallu. Použité příkazy zapisujte bez odřádkování (odřádkování je použito, aby se daný příkaz zobrazil v dokumentu správně). K realizaci tohoto úkolu bude potřeba následující tabulka, která obsahuje IP adresy jednotlivých PC.

Tab. B.1: IP adresy

Hosté	IP adresy
H1	192.168.1.1
H2	192.168.1.2
H3	192.168.1.3
H4	192.168.1.4
H5	192.168.1.5
H6	192.168.1.6
H7	192.168.1.7
H8	192.168.1.8
H9	192.168.1.9

1. Na obou virtuálních strojích zapněte příkazové řádky (černá tabulka na boční liště)
2. Na VM s názvem Mininet přejděte do složky scripts pomocí příkazu:

```
cd scripts/
```

3. Vytvořte topologii příkazem:

```
sudo ./firewallTopology.py
```

a pokud bude vyžadováno zadejte heslo student.

4. V mininetu se příkazem links podívejte na zapojení topologie.
5. Přepněte se do VM Kontrolér.
6. Zde spusťte modul firewall příkazem:

```
$ sudo ryu-manager --verbose
ryu/ryu/app/rest_firewall.py
```

a pokud bude třeba zadejte heslo student

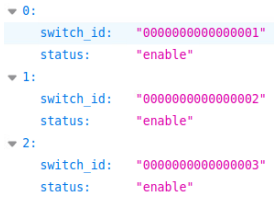
7. Tímto jste propojili topologii s kontrolérem Ryu a jednotlivé přepínače dostali funkci firewallu
8. Nyní otevřete druhý příkazový řádek, kde bude konfigurovat firewall.
9. Pro povolení firewallu na všech přepínačích použijte příkaz:

```
curl -X PUT
http://localhost:8080/firewall/module/enable/all
```

Místo parametru all lze použít SwitchID daného přepínače, čímž bude povolen firewall jen na tomto přepínači. Ověření, že firewall běží zjistíte v grafickém rozhraní Ryu a to spuštěním prohlížeče a zadáním adresy:

```
localhost:8080/firewall/module/status
```

10. Zde by se vám měla zobrazit následující tabulka



Obr. B.12: Spuštění Firewallu

11. Přepněte se zpět do příkazové řádky a použijte klávesu enter, aby jste mohli pokračovat dalšími příkazy.
12. Nyní se seznámíte s konfigurací firewallu.

Konfigurace

Nyní přejdeme ke konfiguraci firewallu. Nejprve se seznámíte s používání jednotlivých příkazů a podle návodu nakonfigurujete pravidla firewallu. Následně samostatně nakonfigurujete zbylá pravidla podle následující tabulky (červená pravidla značí samostatnou práci a zelená jsou pravidla nastavená podle návodu):

Tab. B.2: Pravidla Firewallu

Pravidla	Hosté	
TCP	H2 a H5	H5 a H6
UDP	H1 a H3	H10 a H1
ICMP	H1 a H2	H4 a H6

1. Přejdeme ke konfiguraci pravidel firewallu, bude použit příkazový řádek, který slouží ke konfiguraci těchto pravidel. Ze základu jsou všechny pravidla zakázány a příkazy používanými v následujících úkolech jsou povolovány.
2. Jak je patrné z tabulky B.2, tak nejprve povolíte ICMP mezi hosty h1, h2 a h4 h6 (je nutné vždy pravidlo povolit na obou stranách pokud chceme oboustrannou komunikaci). Docílíme toho použitím následujícího sledu příkazů:

```
Pravidlo pro povolení ICMP mezi h1 a h2
===========================================
curl -X POST -d '{"nw_src": "192.1688.1.1",
"nw_dst": "192.168.1.2", "nw_proto": "ICMP"}'
http://localhost:8080/firewall/rules/0000000000000001

curl -X POST -d '{"nw_src": "192.1688.1.2",
"nw_dst": "192.168.1.1", "nw_proto": "ICMP"}'
http://localhost:8080/firewall/rules/0000000000000001

Pravidlo pro povolení ICMP mezi h4 a h6
===========================================
curl -X POST -d '{"nw_src": "192.1688.1.4",
"nw_dst": "192.168.1.6", "nw_proto": "ICMP"}'
http://localhost:8080/firewall/rules/0000000000000002

curl -X POST -d '{"nw_src": "192.1688.1.6",
"nw_dst": "192.168.1.4", "nw_proto": "ICMP"}'
http://localhost:8080/firewall/rules/0000000000000002
```

Příkaz curl se využívá k přesunu dat, v našem případě jde o přesun dat z kontroléru na daný přepínač, parametr -X říká, že bude použit proxy server a za -d následují složené závorky a v nich data, která chceme poslat. Data jsou zapsána ve formátu json a vypadají následujícím způsobem:

```
curl -X POST -d '{"nw_src": "(zdrojová adresa bez masky)",
"nw_dst": "(cílová adresa bez masky)",
"nw_proto": "(protokol, ICMP,ICMPv6,UDP,TCP)"}'
http://localhost:8080/firewall/rules/(switchID)/(vlan)
```

SwitchID je v našem případě 0000000000000001, což značí přepínač S1.

3. Dalším krokem je oboustranné povolení TCP mezi hostem h2 a h5.

```
Pravidlo pro povolení TCP mezi h2 a h5
==============================================
curl -X POST -d '{"nw_src": "192.1688.1.2",
"nw_dst": "192.168.1.5", "nw_proto": "TCP"}'
http://localhost:8080/firewall/rules/0000000000000002

curl -X POST -d '{"nw_src": "192.1688.1.5",
"nw_dst": "192.168.1.2", "nw_proto": "TCP"}'
http://localhost:8080/firewall/rules/0000000000000002

curl -X POST -d '{"nw_src": "192.1688.1.2",
"nw_dst": "192.168.1.5", "nw_proto": "TCP"}'
http://localhost:8080/firewall/rules/0000000000000001

curl -X POST -d '{"nw_src": "192.1688.1.5",
"nw_dst": "192.168.1.2", "nw_proto": "TCP"}'
http://localhost:8080/firewall/rules/0000000000000002
```

Povšimněte si změny SwitchID, když je prováděna konfigurace hosta umístěného na jiném přepínači.

4. Pravidla, která jste teď nastavili si je možné prohlédnout v grafickém rozhraní na adrese:

```
localhost:8080/firewall/rules/(SwitchID)
```

5. Komunikaci si ověříme na VM Mininet, kde odesláním ICMP zprávy mezi hosty přepínače S1, kde byli pravidla nastavena zjistíme jestli komunikace funguje. ICMP zprávu lze v mininetu mezi hosty poslat následujícím příkazem:

   ```
   <host> ping <host>
   ```

   TCP propojení lze ověřit spuštěním příkazového řádku daného hosta příkazem:

   ```
   xterm <host>(například h1)
   ```

   Spuštění TCP serveru a na druhém hostu spuštěním TCP klienta. Mezi hostem h2 a h5 to provedete příkazem:

   ```
   TCP:
   h2: iperf -s
   h5: iperf -c 192.168.1.2
   ```

6. Dále si zobrazte jednotlivá pravidla v OpenVswitchi spuštěním dalšího příkazového řádku a zadání příkazu:

   ```
   sudo ovs-ofctl dump-flows <přepínač(s1, s2...)>
   ```

7. Vaším samostatným úkolem bude nakonfigurovat pravidla firewallu pro zbylé položky v tabulce. Výsledek vaší práce ukažte vyučujícímu.
8. Po dokončení úkolu vypněte modul Firewall a zavřete topologii.

Otázky

1. Co je zapotřebí udělat jako první po spuštění modulu firewall?
2. Jakým způsobem jsou zadávána pravidla Firewallu?
3. Jak se provádí tvorba pravidla firewallu pokud se dvě PC nachází každé v jiné síti?

B.8.2 Úkol 3: Konfigurace směrování na L3 přepínači pomocí kontroléru

Příprava prostředí

V této části si připravíte prostředí pro následnou konfiguraci směrování L3 přepínačů. Použité příkazy zapisujte bez odřádkování (odřádkování je použito, aby se daný příkaz zobrazil v dokumentu správně). Aby bylo možné provést směrování bude použita následující tabulka IP adres:

Tab. B.3: Směrovací tabulka

Směrovače	Rozhraní	IP adresy
S1	Brána:	192.168.1.1/24
	eth.1:	192.168.10.10/24
S2	Brána:	192.168.2.1/24
	eth.1:	192.168.10.1
	eth.2:	192.168.30.1
	eth.3:	192.168.40.1
S3	Brána:	192.168.3.1/24
	eth.1:	192.168.30.30/24
S4	Brána:	192.168.4.1/24
	eth.1:	192.168.40.40

1. Na obou virtuálních strojích zapněte příkazové řádky (černá tabulka na boční liště)
2. Na VM s názvem Mininet přejděte do složky scripts pomocí příkazu:

```
cd scripts/
```

3. Vytvořte topologii příkazem:

```
sudo ./RouterTopo.py
```

   a pokud bude vyžadováno zadejte heslo student.
4. V mininetu se příkazem links podívejte na zapojení topologie.
5. Přepněte se do VM Kontrolér.
6. Zde spusťte modul Router příkazem:

```
$ sudo ryu-manager --verbose
ryu/ryu/app/rest_router.py
```

   a pokud bude třeba zadejte heslo student
7. Tímto jste propojili topologii s kontrolérem Ryu a jednotlivé přepínače dostali funkci L3 přepínače.
8. Nyní otevřete druhý příkazový řádek, kde bude konfigurovat směrování jednotlivých L3 přepínačů.
9. Nyní se seznámíte s konfiguraci statického směrování L3 přepínačů pomocí kontroléru.

Konfigurace

Přesuneme se konfiguraci směrování jednotlivých L3 přepínačů. Nejprve si procvičíte konfiguraci podle návodu a následně budete mít za úkol směrování dokončit, aby bylo možné komunikovat se všemi hosty. Topologie, kterou bude používat pro směrování je zobrazena na obr. 6.4.

1. Přepněte se do VM Kontrolér.
2. Prvním úkolem, který je třeba udělat je přidání jednotlivých IP adres L3 přepínačům. Jedná se o IP adresy rozhraní, kterými přepínač komunikuje s ostatními a IP adresu přepínače, která funguje jako brána pro hosty v síti. IP adresy se v Ryu přidávají příkazem:

   ```
   curl -X POST -d '{"address":"IP adresa/maska"}'
   http://localhost:8080/router/SwitchID
   ```

   Kupříkladu, když na přepínači S1 přidáme adresu brány a rozhraní, kterým přistupuje k přepínači S2, tak použijeme následující příkazy:

   ```
   curl -X POST -d '{"address":"192.168.1.1/24"}'
   http://localhost:8080/router/0000000000000001

   curl -X POST -d '{"address":"192.168.10.10/24"}'
   http://localhost:8080/router/0000000000000001
   ```

3. Abychom umožnili komunikaci mezi hosty S1 a S2 musí být přidány IP adresy i jeho rozhraní a brány. K tomu použijeme následující příkazy:

   ```
   curl -X POST -d '{"address":"192.168.2.1/24"}'
   http://localhost:8080/router/0000000000000002

   curl -X POST -d '{"address":"192.168.10.1/24"}'
   http://localhost:8080/router/0000000000000002
   ```

4. Přidání IP adres rozhraní do jejich tabulky, ale samo o sobě nezajistí jejich konektivitu. K tomuto je třeba využít nastavení výchozí routy nebo statické routy. V tomto případě postačí výchozí routa hranice sítě, protože přepínač S1 má jedinou možnou cestu směrem k S2. Přepínač S2 má sice více možných cest, ale pokud mu nastavíme tuto cestu (routu) jako výchozí, tak pakety, pro které nebude mít záznam bude posílat touto cestou. Toto nastavení se provádí příkazy:

```
curl -X POST -d '{"gateway":"192.168.10.1"}'
http://localhost:8080/router/0000000000000001

curl -X POST -d '{"gateway":"192.168.10.10"}'
http://localhost:8080/router/0000000000000002
```

Obecně je za gateway dosazena IP adresa výchozí brány, která je pro hosty S1 právě 192.168.10.1 a pro hosty S2 192.168.10.10, protože jsme řekli, že toto bude výchozí routa.

5. Konektivita mezi hosty přepínače S1 a S2 je již nakonfigurovaná. Teď již zbývá nastavit adresy ostatních přepínačů a cestu k jejich hostům. Tato konfigurace se dělá pomocí tzv statických rout, tím nastavíme pevnou cestu do sítě daného přepínače. V Ryu se toto nastavení dělá příkazem:

```
curl -X POST -d '{"destination":"IP adresa sítě/maska",
"gateway": "IP adresa rozhraní"}'
http://localhost:8080/router/SwitchID
```

6. Vaším úkolem bude nastavit zbytek IP adres na přepínačích a jejich statické routy. K dispozici máte topologii dané sítě a jednotlivé adresy. Komunikaci poté můžete ověřit na Mininetu příkazem ping, který byl používán v předchozím úkolu.
7. Vyučujícímu poté ukažte výsledek pomocí příkazu pingall, který odešle zprávu ICMP mezi všemi dvojicemi hostů.
8. Po dokončení úlohy vypněte modul Router a zavřete topologii.

Otázky

1. Jaký je první krok při konfiguraci směrování pomocí Ryu?
2. Jakým způsobem je odesílána konfigurace směrování?
3. Jaký typ směrování je v úloze použit?