

ŽILINSKÁ UNIVERZITA V ŽILINE

FAKULTA RIADENIA A INFORMATIKY

DIPLOMOVÁ PRÁCA

Študijný program:

Aplikované sietové inžinierstvo

Bc. Martin Krška

Softvéroovo-definované siete

Vedúci práce: doc. Ing. Pavel Segeč PhD.

Reg. číslo: 230/2014 Máj 2015

Žilina, 2014

ZADANIE TÉMY DIPLOMOVEJ PRÁCE.

Študijný program : Aplikované sieťové inžinierstvo

Zameranie: Sieťová infraštruktúra

Meno a priezvisko

Martin Krška

Osobné číslo

554726

Názov práce v slovenskom aj anglickom jazyku

Softvérovo-definované siete

Software-defined networks

Zadanie úlohy, ciele, pokyny pre vypracovanie

(Ak je málo miesta, použite opačnú stranu)

Cieľ diplomovej práce:

Na základe oboznámenia sa s SDN navrhnúť a pilotne zrealizovať zapojenie SDN/OpenFlow siete s ukázkou architektúry, protokolov a princípov fungovania v prostredí so simulátorm miniNet a s reálnym sieťovým zariadením realizovaným ako vstavaný počítač.

Obsah:

Pri riešení sa zamerajte:

- Analýza súčasnej technológie SDN, princípov, architektúry a protokolov s ohľadom na aktuálny stav.
- OpenFlow protokol a jeho fungovanie.
- Stanovenie cieľa a vytýčenie funkcií riešenia.
- Analýza a výber modelu vstavaného počítača pre realizáciu platformy OpenFlow prepínača. Pri výbere analyzujte aj iné možnosti ako NetFPGA a programovoľné sieťové procesory.
- Analýza a výber softvérových komponentov SDN/OpenFlow riešenia. Zahŕňa výber vhodnej distribúcie OS Linux, výber kontroléra (zvážte použitie riešenia OpenDaylight) a openFlow prepínača.
- Ako alternatívou reálneho SDN/Openflow riešenia realizujte aj riešenie s využitím sieťového emulátora mininet.
- Pri riešení analyzujte aj možnosti rozšírenia funkcionality prostredníctvom programovania. Po dohode s vedúcim realizujte príklady programového rozšírenia.
- Overte, zdokumentuje a vyhodnotte riešenie.
- V závere navrhnite možnosti ďalšieho skúmania SDN/OpenFlow na KIS s využitím vo vyučovaní.

Témy z predmetov študijného zamerania

5SI041: 2, 5, 7

Meno a pracovisko vedúceho DP:

doc. Ing. Pavel Segeč, PhD., KIS, ŽU

Meno a pracovisko tútora DP:

31.10.2014 JEG

vedúci DP
(dátum a podpis)

tútor
(dátum a podpis)

31.10.2014 JEG - Aténk

vedúci katedry
(dátum a podpis)

Aténk

garant
(dátum a podpis)

Čestné Prehlásenie

Prehlasujem, že som túto bakalársku prácu vytvoril sám a že som uviedol všetky zdroje použité pri jej vytváraní. Súhlasím so zverejnením práce a jej výsledkov.

.....

.....

V Žiline, dňa

Martin Krška

Pod'akovanie

Chcel by som sa pod'akovať vedúcemu bakalárskej práce, doc. Ing. Pavlovi Segečovi PhD., za prípomienky, odbornú pomoc a usmerňovanie pri tvorbe tejto práce.

ABSTRAKT

KRŠKA, Martin: Softvérovo-definované siete

[diplomová práca] - Žilinská univerzita v Žiline. Fakulta riadenia a informatiky; Katedra informačných sietí. - Vedúci: doc. Ing. Pavel Segeč, PhD. - Stupeň odbornej kvalifikácie: Inžinier v študijnom programe Aplikované Sieťové Inžinierstvo - Sieťová Infraštruktúra. Žilina: FRI ŽU v Žiline, 2015. - 85 s.

Cieľom diplomovej práce bolo, na základe oboznámenia sa s SDN sietami, navrhnúť a pilotne zrealizovať zapojenie SDN siete za pomoci protokolu OpenFlow s ukážkou architektúry, protokolov a princípov fungovania v prostredí s emulátorom Mininet a s reálnym sieťovým zariadením realizovaným ako vstavaný počítač. V teoretickej časti práce sú popísané dôvody potreby a princíp SDN sietí, ako aj protokol OpenFlow, ktorý je súčasťou SDN riešenia. V ďalšej časti sú analyzované a následne vybrané vhodné prvky za komponenty SDN siete. V praktickej časti je popis inštalácie, konfigurácie a testovania zhodeného riešenia SDN siete.

Kľúčové slová: Softvérovo-definované siete (SDN). OpenFlow.

ABSTRACT

KRŠKA, Martin: Software-defined networks

[master's thesis] - The University of Žilina. Faculty of Management Science and Informatics; Department of InfoComm Networks. - Tutor: doc. Ing. Pavel Segeč, PhD. - Qualification level: Master in study program Applied Network Engineering - Network Infrastructure. Žilina: FRI ŽU in Žilina, 2013. - 85 p.

The goal of this master's thesis was to, based on the familiarization with SDN networks, design and realize a pilot implementation of SDN network with the help of OpenFlow protocol. Show it's architecture, protocols and workings in the environment of Mininet emulator and with real network device realized as an embedded device. In the theoretical part of work, the reasons for the need and main idea behind SDN networks are described, as well as protocol Openflow, which is a part of the SDN solution. In the next part, appropriate elements are chosen based on the analysis for the SDN components. In the practical part is a description of installation, configuration and testing of the made SDN network solution.

Key words: Software-Defined Networks (SDN). OpenFlow.

Obsah

Zoznam obrázkov	8
Zoznam tabuliek	9
Zoznam skratiek a značiek	10
Úvod	12
1 Softvérovo-definované siete	13
1.1 Dôvod potreby SDN	13
1.2 Problémy súčasných sietí	13
1.3 Princíp SDN	14
1.3.1 Modularita SDN	14
1.3.2 SDN u poskytovateľov pripojenia	17
1.4 Aktuálny stav SDN	18
1.5 Prípady použitia SDN	19
2 OpenFlow	22
2.1 Vznik protokolu OpenFlow	22
2.2 Princíp fungovania protokolu	22
2.3 OpenFlow porty	24
2.3.1 Štandardné porty	25
2.3.2 Fyzické porty	25
2.3.3 Logické porty	25
2.3.4 Rezervované porty	26
2.4 OpenFlow tabuľky	26
2.4.1 Zretežené spracovanie	26
2.4.2 Tabuľka tokov	27
2.4.3 Porovnávanie	28
2.4.4 Odstránenie toku	29
2.4.5 Tabuľka skupín	29
2.5 OpenFlow komunikácia	29
2.5.1 Správy kontrolér-prepínač	30
2.5.2 Asynchronne správy	30
2.5.3 Symetrické správy	31
2.6 Verzie protokolu OpenFlow	31

3 Ciele práce	34
4 Analýza a výber SDN komponentov	35
4.1 Analýza vstavaných počítačov	35
4.1.1 Cubieboard1	35
4.1.2 Cubieboard2	36
4.1.3 Banana Pi	36
4.1.4 Raspberry Pi model B+	37
4.1.5 Vybraný produkt	38
4.2 NetFPGA a programovateľné sieťové procesory	39
4.2.1 NetFPGA	39
4.2.2 Štandardný prepínací hardvér	39
4.2.3 Špecializovaný prepínací hardvér	41
4.2.4 Programovateľné sieťové procesory	41
4.3 Analýza softvérových prepínačov	42
4.3.1 Indigo Virtual Switch	42
4.3.2 Open vSwitch	42
4.3.3 Vybraný produkt	43
4.4 Analýza SDN kontrolérov	43
4.4.1 Floodlight	43
4.4.2 Ryu	44
4.4.3 ONOS	44
4.4.4 OpenDaylight Helium	44
4.4.5 HP VAN SDN kontrolér	46
4.4.6 Vybraný produkt	47
5 Ukážky infraštruktúry SDN siete	48
5.1 Virtuálna sieť	48
5.1.1 Mininet	48
5.1.2 Inštalácia emulátora	49
5.1.3 Práca s emulátorom	49
5.1.4 Vytvorenie vlastnej topológie	51
5.2 Fyzická sieť	52
5.2.1 Banana Pi	52

5.3	Zmiešaná siet	54
5.4	Práca s tokmi v OpenFlow prepínači	56
6	Kontrolér OpenDaylight Helium	59
6.1	Základné nastavenie	59
6.2	Funkcionalita protokolu OpenFlow	60
6.2.1	Počiatočná komunikácia	60
6.2.2	Tabuľky tokov prepínača	61
6.3	Napojenie ukážkových sietí na kontrolér	63
6.4	Web rozhranie kontroléra	64
6.4.1	YANG UI	65
6.5	Práca s tokmi cez kontrolér	66
6.5.1	Postman	66
6.5.2	Posielanie požiadaviek cez REST API	66
6.5.3	Blokovanie IPv4 komunikácie	67
6.5.4	Výpis tokov cez kontrolér	70
6.5.5	Odstránenie tokov cez kontrolér	70
6.5.6	Blokovanie ICMP správ	70
6.6	Clustering kontroléra	72
6.6.1	Clustering na jednom kontroléri	73
6.6.2	Clustering na viacerých kontroléroch	73
6.7	Rozšírenie funkcionality kontroléra	74
6.7.1	Technický prehľad architektúry kontroléra	74
6.7.2	Service Abstraction Layer	75
6.7.3	Možnosti programovania kontroléra	77
6.7.4	API Explorer	78
7	Záver	80
Literatúra		84
Príloha A		85

Zoznam obrázkov

1	Modularita riadiacej úrovne SDN	15
2	Ukážka abstrakcie sietovej topológie	15
3	SDN architektúra [1, strana 7]	16
4	Hlavné komponenty OpenFlow prepínača [5, strana 8]	23
5	Tok paketov cez zretežené spracovanie [5, strana 15]	26
6	Diagram popisujúci prechod paketu cez prepínač [5, strana 17]	28
7	Cubieboard1 [12]	35
8	Cubieboard2 [13]	36
9	Banana Pi [15]	37
10	Raspberry Pi model B+ [17]	37
11	OpenDaylight Helium diagram [31]	45
12	Topológia virtuálnej siete	51
13	Topológia fyzickej siete	54
14	Topológia zmiešanej siete	55
15	Jednoduchá topológia	56
16	Úvodná výmena OpenFlow správ	61
17	DLUX topológia virtuálnej siete z časti 5.1.4	65
18	DLUX modul YANG UI	66
19	Tvorenie HTTP správ v prostredí Postman	69
20	Service Abstraction Layer [38]	76
21	Data Packet Service [38]	76
22	API Explorer	79

Zoznam tabuliek

1	Tabuľka tokov	27
2	Tabuľka skupín	29
3	Porovnanie vstavaných počítačov	38
4	Zoznam užitočných OpenDaylight Java rozhraní	75

Zoznam skratiek a značiek

ACL - Access Control List
API - Application Programming Interface
ARP - Address Resolution Protocol
ASIC - Application-Specific Integrated Circuit
BGP - Border Gateway Protocol
CLI - Command Line Interface
DDoS - Distributed Denial of Service
DLUX - OpenDaylight User Experience
DNS - Domain Name System
EPL - Eclipse Public Licence
FPGA - Field Programmable Gate Array
GRE - Generic Routing Encapsulation
GUI - Graphical User Interface
HTTP - Hypertext Transfer Protocol
IDS - Intrusion Detection System
IP - Internet Protocol
IPS - Intrusion Prevention System
IPsec - Internet Protocol Security
ISP - Internet Service Provider
IVS - Indigo Virtual Switch
JSON - JavaScript Object Notation
JVM - Java Virtual Machine
KVM - Kernel-based Virtual Machine
LISP - Locator/ID Separation Protocol
LLDP - Link Layer Discovery Protocol
MAC - Media Access Control
MPLS - Multiprotocol Label Switching
NAT - Network Address Translator
NETCONF - Network Configuration Protocol
NFV - Network Function Virtualization
NPS - Network Processors for Smart networks

OF-CONFIG - OpenFlow Management and Configuration Protocol

ONF - Open Networking Foundation

ONIE - Open Network Install Environment

ONOS - Open Networking Operating System

OS - Operating System

OSGi - Open Service Gateway initiative

OSI - Open Systems Interconnection

OVSDB - Open vSwitch Database Management Protocol

PBB - Provider Backbone Bridging

PCI - Peripheral Component Interconnect

PoP - Point of Presence

PPP - Point-to-Point Protocol

QoS - Quality of Service

RAM - Random-Access Memory

REST - REpresentational State Transfer

SAL - Service Abstraction Layer

SDN - Software Defined Networks

SNMP - Simple Network Management Protocol

SoC - System on a Chip

TCAM - Ternary Content Addressable Memory

TCP - Transport Control Protocol

TLS - Transport Layer Security

TLV - Type-Length-Value

VLAN - Virtual Local Area Network

VM - Virtual Machine

VPN - Virtual Private Network

VXLAN - Virtual Extensible Local Area Network

WAN - Wide Area Network

XMPP - Extensible Messaging and Presence Protocol

ICMP - Internet Control Message Protocol

Úvod

Rapídný nárast mobilných zariadení a ich obsahu, serverovej virtualizácie a nástup clou-dových služieb patrí medzi hlavné hnacie sily, ktoré nútia odvetvie počítačových sietí zno-va sa zamyslieť nad tradičnou architektúrou sietí. Architektúra terajších sietí nie je vhodná pre splnenie požiadaviek novodobých spoločností, poskytovateľov internetového pripoje-nia (ISP) ani koncových používateľov.

Softvérovo-definované siete (SDN) sú vznikajúcou architektúrou, ktorá je dynamická, ovlá-dateľná, efektívna a prispôsobivá. Vďaka tomu je ideálna pre vysoko kapacitnú, dynamickú povahu súčasných aplikácií. Táto architektúra oddeluje riadenie siete od transportných funkcií, čím umožňuje priamu programovateľnosť riadiacej časti a abstrakciu aplikácií a sieťo-vých služieb infraštruktúry ležiacej pod ňou.

Cieľom tejto práce je pilotne zrealizovať zapojenie a ukážku fungovania SDN siete založe-nej na protokole OpenFlow. V práci som sa zameral na prieskum súčasných SDN technológií a popis protokolu OpenFlow. Preskúmal som rôzne modely vstavaných počítačov vhodných pre implementovanie SDN technológie a vybral jeden pre realizáciu praktickej ukážky SDN siete. Zanalyzoval som a vykonal výber vhodných softvérových komponentov pripravovanej SDN siete. Súčasne som sa venoval emulátoru Mininet určenému na tvorbu SDN sietí a pre-skúmal som možnosti ich programovania. V hlavnej časti som vykonal inštaláciu, konfigurá-ciu a testovanie zhodeného riešenia SDN siete a na záver zhodnotil výsledky mojej práce.

1 Softvérovo-definované siete

1.1 Dôvod potreby SDN

Veľa bežných sietí je hierarchických, postavených s rádmi Ethernetových prepínačov usporiadaných do stromovej štruktúry. Tento dizajn mal zmysel, keď prevládal model klient-server aplikácií. Avšak takáto statická architektúra nevyhovuje dynamickým výpočtom a úložným potrebám dátových centier novodobých podnikov, kampusom a prostrediam prenosových sietí. Niektoré z hlavných podnetov pre zmenu v počítačových sieťach sú [1]:

- Zmena tvarov dátovej prevádzky.
- Splývanie osobného a firemného použitia IT technológií.
- Nástup clouдовých služieb.
- Potreba väčšej šírky pásma pre zaobchádzanie s objemnými dátami.

Existujúce sieťové architektúry neboli navrhnuté pre splnenie požiadaviek súčasných používateľov, podnikov a ISP. Sieťoví architekti sú obmedzený možnostami súčasných sietí, ktoré zahŕňajú:

- Komplexnosť vedúcu ku stagnácii.
- Inkonzistencia politík.
- Náročnosť rozširovania siete.
- Závislosť od predajcu sieťových zariadení.

1.2 Problémy súčasných sietí

Existuje niekoľko pretrvávajúcich problémov so súčasnými sieťami [2]. Medzi hlavné patrí:

- Správa siete je náročná.
- Siete sa ťažko vyvíjajú.
- Dizajn sietí nie je postavený na formálnych princípoch (chýba štandardizácia)

V nasledujúcej časti pojmom preposielanie označujem prenos paketu zo vstupného portu zariadenia na jeho výstupný port na základe tabuľky tokov (preposielacieho stavu). Touto tabuľkou sa myslí smerovacia tabuľka optimalizovaná pre vyhľadávanie a obsahujúca dodatočné informácie ku jednotlivým položkám, ako napríklad cieľová Media Access Control (MAC) adresa a pod.

Logika sietí je obsiahnutá v dvoch úrovniach:

- Dátová úroveň - má na starosti spracovanie príchodzích paketov v rámci sietového zariadenia (lokálne), kde na základe nastaveného preposielacieho stavu a hlavičky príchodzieho paketu oňom vykoná rozhodnutie.
- Riadiaca úroveň - má na starosti výpočet preposielacieho stavu daného zariadenia. Na tento výpočet sa väčšinou využívajú smerovacie protokoly, manuálna konfigurácia a pod. Pre tieto výpočty potrebuje mať zariadenie prehľad o svojom okolí.

Dátová úroveň je realizovaná pomocou sady abstrakcií, ktoré poznáme ako vrstvy sietového modelu Open Systems Interconnection (OSI). Vďaka tomu sa problém rozložil do menších kúskov (vrstiev), čo umožnilo rýchle inovovanie v rámci jednotlivých vrstiev.

Oproti tomu riadiaca úroveň nemá takúto formu abstrakcie. Existuje tam niekoľko mechanizmov pre dosiahnutie rôznych cieľov:

- Smerovanie - distribučné smerovacie algoritmy.
- Izolovanie - Access Control List (ACL), Virtual Local Area Network (VLAN), Firewally a pod.
- Traffic engineering - Multiprotocol Label Switching (MPLS), úprava váh a pod.

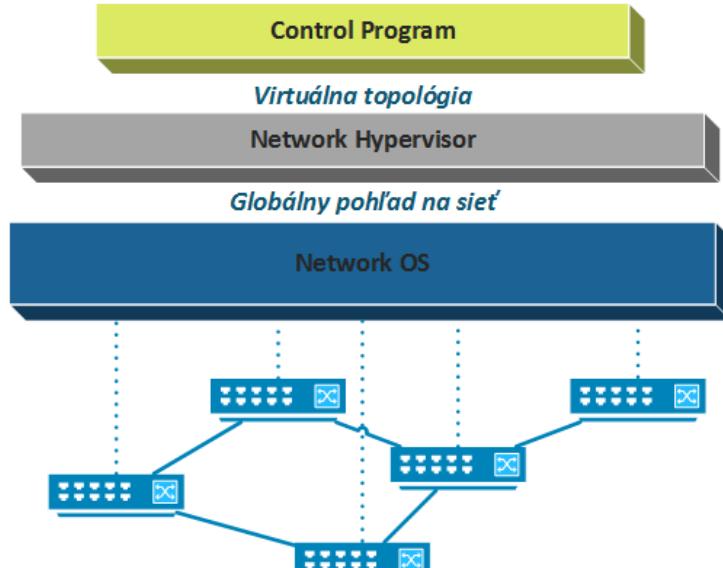
Pretože sú ale tieto mechanizmi postavené bez abstrakcie, chýba tu modularita, kvôli ktorej je nutné vyvíjať nové mechanizmy vždy od začiatku, čo má za následok ich obmedzenú funkcionality. Toto bol taktiež hlavný dôvod, pre ktorý bolo SDN vyvinuté.

1.3 Princíp SDN

1.3.1 Modularita SDN

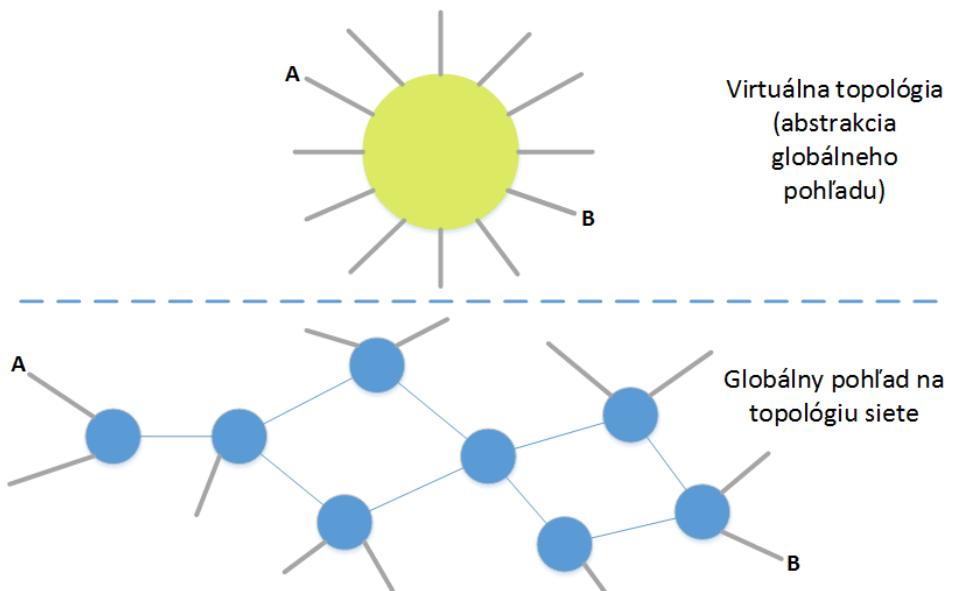
Princípom SDN je použitie abstrakcie na problémy, ktoré musí riešiť riadiaca úroveň a teda rozdelenie týchto problémov do viacerých vrstiev/komponentov [2], ktoré sú:

- Network Operating System (Network OS) - poskytuje globálny pohľad na topológiu siete (komunikuje so sietovými zariadeniami). Takisto pozoruje a riadi ich dátovú úroveň.
- Network Hypervisor - poskytuje virtuálnu topológiu (abstrakciu sietovej topológie) pre jednoduchšiu aplikáciu zmien priyatých od vyššej vrstvy.
- Control Program - aplikácie vyjadrujúce ciele nad virtuálnou topológiou siete.



Obr. 1: Modularita riadiacej úrovne SDN

Sieťová virtualizácia sa dá pokladať za hlavnú výhodu SDN sietí oproti tradičným sieťam. V dnešnej dobe sa stali cloudové riešenia veľmi populárne a veľa spoločností migruje svoje siete do týchto prostredí. Pri presúvaní svojej siete do cloutu si ale spoločnosť chce zachovať svoju politiku nad existujúcou topológiou. V takýchto prípadoch sa dá využiť schopnosť SDN virtualizovať sieť. SDN umožní špecifikovať cloutu virtuálnu topológiu, čím umožní plynulú migráciu do alebo zo súčasnej siete.



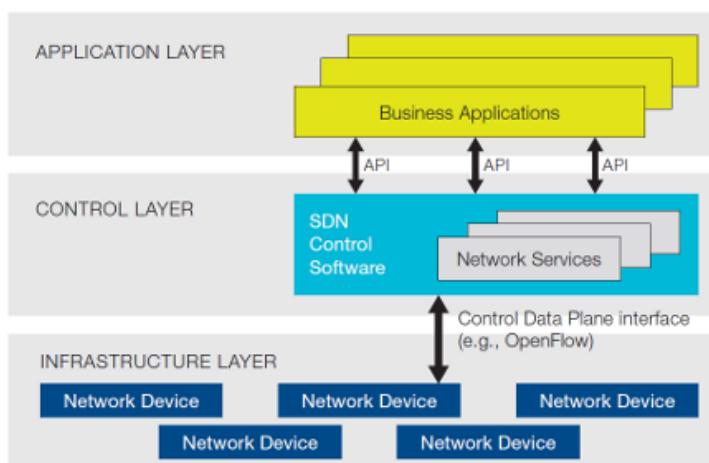
Obr. 2: Ukážka abstrakcie sieťovej topológie

SDN je teda architektonický prístup, ktorý optimalizuje a zjednoduší sieťové operácie vytvorením silnejšej väzby medzi interakciami aplikácií, sieťovými službami a zariadeniami, či už reálnymi alebo virtuálnymi [3, strana 7].

Toto je často dosiahnuté využitím bodu logickej centralizovanej sietovej kontroly (centralizáciou riadiacej úrovne), ktorý sa nazýva SDN kontrolér. Ten organizuje, sprostredkováva a napomáha komunikácii medzi aplikáciami, ktoré si želajú interakciu so sietovými prvkami a sietovými prvkami, ktoré chcú týmto aplikáciám posielat informácie. SDN kontrolér je hlavným prvkom v modeli SDN siete.

Za hlavné vrstvy v modeli SDN sa považujú:

- Vrstva infraštruktúry - obsahujúca sietové zariadenia, ktorých dátovú úroveň ovláda SDN kontrolér danej siete. Zariadenia s SDN kontrolérom komunikujú cez spoločné rozhranie, z ktorých najznámejšie na to určené sa nazýva OpenFlow.
- Riadiaca vrstva - obsahujúca SDN kontrolér komunikujúci so zariadeniami v sieti a zodpovedajúci za stav ich dátovej úrovne, ktorý nastavuje podľa požiadaviek jednotlivých aplikácií.
- Aplikačná vrstva - obsahujúca používateľské aplikácie komunikujúce svoje požiadavky na siet.



Obr. 3: SDN architektúra [1, strana 7]

Na komunikáciu SDN kontroléra so sietovými prvkami a s aplikáciami určenými na riadenie siete sa využíva Application Programming Interface (API), teda rozhranie pre programovanie aplikácií. Tieto API sa ďalej rozdeľujú na:

- Southbound API - slúžiaci na komunikáciu medzi SDN kontrolérom a jednotlivými zariadeniami v sieti.
- Northbound API - slúžiaci na komunikáciu medzi SDN kontrolérom a aplikáciami na riadenie siete.

SDN sa často navzájom zamieňa s Network Function Virtualization (NFV), čo je koncept sietovej architektúry vyvinutý konzorcium poskytovateľov pripojenia (ETSI). Jeho zámerom

je relokácia sietových funkcií z dedikovaných zariadení na generické servery. SDN a NFV sú na sebe nezávislé, ale spoločne sa dopĺňajú a uľahčujú tým svoje ciele [4].

1.3.2 SDN u poskytovateľov pripojenia

V prenosových sietach ISP je potrebné použiť trochu iný prístup ku SDN [2], pretože takéto siete nie sú homogénne. Keď koncové zariadenie posielalo paket do siete, existujú tri logické rozhrania, ktoré pri tom zohrávajú rolu:

- klient-siet[#] - udáva cieľ danej správy. Môže požiadať o Quality of Service (QoS).
- paket-smerovač - vyhľadanie správnej položky v preposielacom stave (tabuľke).
- operátor-siet[#] - nastoľovanie pravidiel v sieti, ako Traffic Engineering, izolovanie, prístupová kontrola a pod.

V tradičných sietach sa hlavička paketu využíva pri rozhraniach klient-siet[#] a paket-smerovač. Neexistuje ale všeobecné rozhranie operátor-siet[#].

V SDN sú stále zmiešané rozhrania klient-siet[#] a paket-smerovač, avšak poskytuje rozhranie operátor-siet[#].

Pre kompletné oddelenie funkcií jednotlivých rozhraní sa odporúča využitie SDN spolu s technológiou MPLS, ktorá odlišuje okraj siete od jadra spôsobom:

- Okraj používajúci hlavičku paketu je rozhranie klient-siet[#], pričom následne pridáva do paketu MPLS značku.
- Jadro používajúce MPLS značku je rozhraním paket-smerovač.

Vďaka tomu sa nám vytvorí prehľadná sietová modularita, v ktorej:

- Okraj používa rozhranie klient-siet[#].
- Jadro používa rozhranie paket-smerovač.
- SDN používa rozhranie operátor-siet[#].

Týmto sa zachová jednoduché jadro a všetká komplexnosť sa posunie na okraj siete, vďaka čomu sa aj zlepší pružnosť jadra. Jadro by v takomto prípade mohlo byť zložené so stávajúcich zariadení, používajúcich Application-Specific Integrated Circuit (ASIC), pre efektívne prepínanie paketov pomocou MPLS. Okraj siete by však bol tvorený softvérovými prepínačmi, používajúcimi univerzálne procesory s architektúrou x86. Aj keď tieto procesory nie sú také rýchle alebo nákladovo efektívne ako čipy ASIC, postačujú na zvládanie prevádzky na

okraji siete pričom ich výhodou je zvýšená flexibilita a nižšia cena. Tieto procesory sa takisto stávajú výkonnejšími príchodom každej novej generácie.

SDN by taktiež mohli implementovať na okrajových prepínačoch funkcionalitu takzvaných middlebox zariadení, teda zariadení, ktoré manipulujú s dátovou prevádzkou v sieti za iným účelom ako je smerovanie paketov. Príkladom takýchto zariadení je napríklad Firewall, Intrusion Detection System (IDS), Intrusion Prevention System (IPS), Network Address Translator (NAT), Load Balancer a pod. Takéto zariadenia už totižto pracujú na okraji sietí a väčšina z nich používa na spracovanie paketov x86 procesory. Toto riešenie by preto bolo logickým krokom vpred.

1.4 Aktuálny stav SDN

Od prijatia myšlienky SDN spoločnosťami vo svete ubehlo len pár rokov, avšak táto idea zaznamenala výraznú explóziu v podobe veľkého množstva SDN riešení ponúkaných rôznymi spoločnosťami a zameraných na riešenie rôznych problémov v oblasti sieťovania [3, 5]. Tento nový trh poskytol priestor pre rozvoj začínajúcim firmám, ako napríklad Nicira (odkúpená firmou VMware), Big Switch Networks a Pica8. Rovnako donútil aj veľké firmy, ako napríklad Cisco, Juniper a Brocade zaujímať sa o SDN technológiu a držať krok s jej vývojom.

SDN podporuje ideu otvorených štandardov, čím poskytuje o niečo väčšiu flexibilitu a otvorenosť aká bola doteraz, kde boli zákazníci odkázaní na jedného predajcu sieťových zariadení ak chceli ich funkcionalitu plnohodnotne využívať. Jedným z dôkazov tohto tvrdenia je fakt, že všetci významní predajcovia sieťových zariadení v nich začali, vo väčšej alebo menšej miere, podporovať protokol OpenFlow. Tento protokol urýchľil nástup SDN, v ktorom zastáva úlohu southbound API. Za hlavný katalyzátor v nástupe SDN sa ale považuje dostupnosť čistého prepínacieho hardvéru bez operačného systému. Jeho nízka cena a možnosť prispôsobenia uľahčuje implementáciu SDN a ponúka konkurenciu ku drahším, proprietárnym produktom predajcov, ako sú Cisco, Juniper a pod.

Takisto existuje veľa spôsobov, ako implementovať SDN do siete. Existujú softvérové prepínače obsahujúce podporu pre SDN, ktoré nepotrebuju špecializovaný hardvér. V súčasnosti je dokonca vo svete nasadených viac softvérových prepínačov ako fyzických. Taktiež niektorí výrobcovia ponúkajú špeciálne navrhnuté hardvérové prepínače, ktoré majú zlepšený výkon v SDN prostredí. Iní pridávajú do svojich tradičných zariadení podporu pre SDN.

Za hlavný prvok rozlišujúci jednotlivé riešenia sa pokladá SDN kontrolér, ktorý je kľúčo-

vým a zároveň najkomplexnejším prvkom SDN. Okrem jeho schopností a efektívnosti sú dôležité aj spôsoby, akými komunikuje so sieťovými prvkami a s aplikáciami. Teda aké southbound a northbound API používa.

Existuje množstvo proprietárnych aj voľne šíriteľných riešení. Niektoré z dostupných SDN kontrolérov sú ONIX, Beacon, Floodlight, OpenDaylight, Ryu, OpenContrail a FlowVisor. Viačeré z týchto kontrolérov majú otvorený kód a slúžia ako stavebné štruktúry, z ktorých návrhu vychádzali firmy pri tvorbe svojich vlastných riešení.

Medzi používané southbound API patria protokoly, ako OpenFlow, Open vSwitch Database Management Protocol (OVSDB), OpFlex, Network Configuration Protocol (NETCONF), Locator/ID Separation Protocol (LISP), Extensible Messaging and Presence Protocol (XMPP), Border Gateway Protocol (BGP), Simple Network Management Protocol (SNMP) a pod. Vývoj u northbound API je stále chaotický a firmy zväčša vytvárajú vlastné rozhrania na komunikáciu medzi SDN kontrolérom a aplikáciami. Jedným z viac používaných rozhraní je REpresentational State Transfer (REST), prevažne spájaný s protokolom Hypertext Transfer Protocol (HTTP).

1.5 Prípady použitia SDN

Spoločnosti predávajúce SDN produkty ponúkajú svojim zákazníkom rôzne prípady použitia SDN pre vyriešenie problémov, ktoré boli v tradičných sietach prekážkou. Vymenované sú niektoré z týchto využití [6].

Sieťová virtuálizácia - siete s viacerými nájomcami (dátové centrá): dynamické vytváranie oddelených topologicky rovnocenných sietí naprieč dátovým centrom so škálovateľnosťou nad limity typických VLAN sietí súčasnosti.

- Dosiahnuté výhody: zlepšené využitie zdrojov dátových centier, pohybujúce sa okolo 20-30%. Rýchlejší spätné časy vo vytváraní oddelených sietí, z týždňov na minúty cez automatizáciu rozhraní API.

Sieťová virtuálizácia - elastické siete (dátové centrá): vytváranie polohovo-agnostických sietí naprieč rackmi alebo dátovými centrami, s mobilitou virtuálnych strojov (VM) a dynamickou realokáciou zdrojov.

- Dosiahnuté výhody: zjednodušené aplikácie, ktoré je možné urobiť viac odolnými bez komplikovaného programovania, lepšie využitie zdrojov, keďže VM sú transparentne premiestňované pre konsolidáciu pracovnej záťaže. Zlepšený čas obnovy pri havariách.

Vkladanie služieb (dátové centrá/ISP/WAN): vytvorenie dynamických zretežení L4-7 služieb pre jednotlivých nájomcov na zaopatrenie samoobslužného výberu L4-7 služieb alebo tých, potrebných vzhľadom na politiku (napríklad zapnutie DDoS ochrany ako odozvy na útok, samoobslužný firewall, IPS služby v hostujúcich prostrediach a pod).

- Dosiahnuté výhody: časy obstarania sú zredukované z týždňov na minúty, zlepšená agilita a samoobsluha umožňujú nové výnosy a príležitosti služieb s podstatne nižšou cenou za službu.

Agregačné čerpanie (dátové centrá/kampusy/prístupové siete): poskytuje viditeľnosť a schopnosti odstraňovania porúch, monitorovania dátovej prevádzky a merania výkonu aplikácií na ľubovoľnom porte v nasadení s viacerými prepínačmi bez použitia početných, drahých monitorovacích zariadení.

- Dosiahnuté výhody: výrazné ušetrenie a redukcia ceny. Menšie režijné náklady pri počiatočnom nasadení, znižovaním potreby pre vedenie nadbytočných káblov z monitorovacích zariadení do každého prepínača.

Dynamické presmerovanie vo WAN - presun veľkého množstva dôveryhodných dát obchádzajúc drahé inšpekčné zariadenia (ISP/Enterprise Edge): poskytuje dynamický, ale zároveň autentifikovaný, programovateľný prístup pre odklon na úrovni tokov použitím rozhraní do sietových prepínačov a smerovačov.

- Dosiahnuté výhody: ušetrenie nákladných, nepotrebných investícií do 10/100Gbps L4-7 firewallov, load-balancerov, IPS/IDS spracúvajúcich nepotrebnú dátovú prevádzku.

Dynamické WAN prepoje (ISP): vytváranie dynamických spojení na Internetových prepojoch medzi linkami viacerých spoločností alebo medzi viacerými ISP využívaním efektívnych, vysoko výkonných prepínačov.

- Dosiahnuté výhody: schopnosť okamžitého spojenia znižuje operačné náklady pre vytváranie prepojov medzi organizáciami, poskytujúc schopnosť na umožnenie samoobsluhy.

Šírka pásma na požiadavku (ISP): umožňuje programovú kontrolu na prenosových linkách pre vyžiadanie extra šírky pásma, keď je potrebná (napríklad pri zálohovaní, pri obnove po havárii a pod.)

- Dosiahnuté výhody: znižuje operačné náklady, umožnením zákazníckej samoobsluhy a zväčšená agilita šetriaca týždne manuálneho poskytovania.

Virtuálny okraj (prístupové siete ISP): v kombinácii s NFV, nahradí existujúce zariadenie na strane zákazníka (CPE) za odľahčenú verziu. Bežné funkcie a komplexné spracovanie prenosov dát sa presunie na PoP (Points-of-Presence) alebo dátové centrum ISP.

- Dosiahnuté výhody: zvyšuje dobu použiteľnosti CPE, zlepšuje schopnosť odstraňovaania porúch, znižuje počet výjazdov ku zákazníkovi a pridáva flexibilitu predaja nových služieb zákazníkovi.

Sietová bezpečnosť: existuje niekoľko spôsobov akými dokáže SDN zlepšiť bezpečnosť siete. Medzi hlavné z nich patria [7]:

- Selektívne blokovanie škodlivej dátovej prevádzky na koncové zariadenia a zároveň stále umožňovať tok normálnej prevádzky. V tomto prípade by SDN bolo spojené so zariadením pre detekciu malvéru.
- Zlepšenie auditov a konfliktov detekcie/rozhodnutia bezpečnostnej politiky. SDN by mohlo byť využité na agregáciu a správu sietovej segmentácie.
- Centralizovanie bezpečnostnej politiky služieb a konfiguračného manažmentu. V tomto prípade by SDN mohlo zosúladiť bezpečnostnú politiku so serverovou virtualizáciou, cloudom alebo platformou na orchestráciu.
- Automatizácia sietovej bezpečnosti, tzv. ‘sebaobranné siete’. Na základe najnovších hrozieb, kombinácia firewallu a SDN kontroléra, môže vygenerovať za behu nové pravidlá pre firewall.
- Implementácia viac špecifikovanej sietovej segmentácie. Takzvaná mikro-segmentácia, kde by jednotliví používateelia, relácie, toky mohli komunikovať cez point-to-point Virtual Private Network (VPN).

2 OpenFlow

2.1 Vznik protokolu OpenFlow

Open Networking Foundation (ONF) je nezisková organizácia zameraná na potreby používateľov a propagáciu adopcie SDN sietí prostredníctvom vývoja otvorených štandardov [8]. bola založená spoločnosťami Deutsche Telekom, Facebook, Google, Microsoft, Verizon a Yahoo! v roku 2011. V dobe písania tejto práce má ONF viac ako 150 členov, medzi ktorími je veľa známych a dôležitých spoločností. ONF si udržiava niekoľko pracovných skupín, ktoré sa zaobrajú vývojom OpenFlow štandardu pre jeho využitie v nových prípadoch a nasadenie v produkčnom prostredí.

Jej hlavným príspevkom svetu sa stal protokol OpenFlow, ktorý je verejnou považovanou za prvý SDN štandard. Tento protokol umožňuje riadiacej úrovni (OpenFlow kontroléru) interakciu s dátovou úrovňou (sietovými zariadeniami) v prostredí SDN. OpenFlow bol vytvorený na univerzite Standford v USA ako časť programu Clean Slate a verejnosti bol predstavený v roku 2008 pre poskytnutie alternatívy ku proprietárny riešeniam, ktoré limitovali flexibilitu a vytvárali závislosť na konkrétnom predajcovi. Špecifikácia bola vyvinutá na umožnenie výskumníkom spúštať experimenty na heterogénnych prepínačoch a smerovačoch v jednotnom spôsobe, bez potreby predajcov zverejniť vnútorné fungovanie ich produktov alebo potreby výskumníkov písat kontrolný softvér pre konkrétnego predajcu.

Jeho popularita spočíva aj v tom, že je to mechanizmus využívajúci Ternary Content Addressable Memory (TCAM) tabuľky v už existujúcich sietových zariadeniach pre svoj koncept programovateľných tokov. Aj preto nevyžaduje pre svoje fungovanie veľa zmien v sietových zariadeniach.

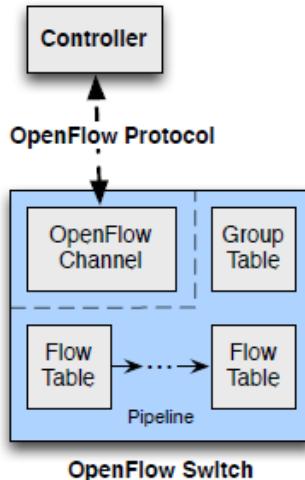
V nasledujúcich častiach popíšem verziu 1.3 tohto protokolu [9], ktorá je v súčasnosti najnovšia z jeho verzií, široko podporovaných v produktoch predajcov sietových zariadení, softvérových prepínačoch a SDN kontroléroch.

2.2 Princíp fungovania protokolu

Špecifikácia definuje dva SDN komponenty:

- OpenFlow kontrolér - jeho úlohou je naprogramovanie OpenFlow tabuľiek v sietových zariadeniach (OpenFlow prepínačoch). Je to SDN kontrolér s podporou OpenFlow.
- OpenFlow prepínač - prijíma paket a zaobchádza s ním v súlade s jeho tabuľkami tokov.

OpenFlow špecifikácia definuje protokol medzi OpenFlow kontrolérom a OpenFlow prepínačmi a množinu operácií nad týmito prepínačmi. Tento protokol komunikuje buď cez Transport Layer Security (TLS) alebo nechránený Transport Control Protocol (TCP).



Obr. 4: Hlavné komponenty OpenFlow prepínača [5, strana 8]

OpenFlow logický prepínač pozostáva z jednej alebo viacerých *flow tables* (tabuľiek tokov) a *group table* (skupinovej tabuľky), ktoré vykonávajú vyhľadávanie a preposielanie paketov. Prepínač má jeden alebo viac *OpenFlow Channel* (OpenFlow kanálov) na externý kontrolér. Prepínač komunikuje s kontrolérom a ten spravuje prepínač cez OpenFlow protokol.

Používaním OpenFlow protokolu, kontrolér vie pridať, upraviť a vymazať *flow entries* (položky toku) v tabuľkách tokov a to reaktívne (ako odozva na pakety) alebo proaktívne. Každá tabuľka tokov v prepínači pozostáva z množiny položiek toku, kde každá položka pozostáva z polí *match fields*, *counters* a sada položiek *instructions* (instrukcie), ktoré sa aplikujú na zhodné pakety.

Porovnávanie začína v prvej tabuľke tokov a môže pokračovať do ďalších tabuľiek tokov cez *pipeline* (zreťazené spracovanie). Položky toku porovnávajú pakety v prioritnom poradí, s použitím prvej zhodnej položky v každej tabuľke. Ak je nájdená zhodná položka, vykonajú sa inštrukcie asociované s konkrétnou položkou toku. Ak nie je nájdená zhoda v tabuľke tokov, výsledok záleží na konfigurácii položky toku *table-miss*: ku príkladu, paket môže byť preposlaný kontroléru cez OpenFlow kanál, zahodí sa, alebo jeho porovnanie môže pokračovať v ďalšej tabuľke tokov.

Inštrukcie asociované s každou položkou toku môžu obsahovať akcie, alebo modifikovať proces zreťazeného spracovania. Akcie zahrnuté v inštrukciách popisujú preposielanie paketu, jeho úpravu a spracovanie skupinovej tabuľky. Inštrukcie zreťazeného spracovania umožňujú paketom byť odoslané do nasledujúcich tabuľiek na ďalšie spracovanie a umož-

ňuje komunikáciu informácií, v podobe metadát, medzi tabuľkami. Zreťazené spracovanie medzi tabuľkami končí, keď inštrukčná sada asociovaná so zhodujúcou sa položkou toku už nešpecifikuje ďalšiu tabuľku. V tomto momente je paket zvyčajne upravený a odoslaný.

Položky toku môžu byť preposlané na *port*. Toto je zvyčajne fyzický port, ale môže to byť aj logický port definovaný prepínačom, alebo rezervovaný port definovaný touto špecifikáciou. Rezervované porty môžu označovať generické preposielacie akcie, ako posielanie kontroléru, flooding, alebo smerovanie inými metódami ako OpenFlow, napríklad tradičným spracovaním prepínačom, zatiaľ čo prepínačom definované logické porty môžu špecifikovať agregičné skupiny, tunely, alebo loopback rozhrania.

Akcie asociované s položkami toku môžu taktiež nasmerovať pakety do skupiny, ktorá špecifikuje dodatočné spracovanie. Skupiny reprezentujú sadu akcií pre flooding, ako aj komplexnejšie preposielacie sémantiky (multipath, fast reroute, link aggregation a pod). Ako celkové pravidlo bezcielenosti, skupiny taktiež umožňujú viacerým položkám toku preposlanie na jednotný identifikátor (napríklad IP preposielanie na spoločný next hop). Táto abstrakcia umožňuje spoločným výstupným akciám naprieč položkami toku byť efektívne zmenených.

Skupinová tabuľka pozostáva z položiek skupiny, pričom každá položka skupiny obsahuje zoznam takzvaných *action buckets* (akčných vedier) so špecifickými sémantikami závislými od typu skupiny.

2.3 OpenFlow porty

OpenFlow porty sú sietovými rozhraniami pre prechod paketov medzi OpenFlow spracovaním a zvyškom siete. Prepínače sa na seba logicky pripájajú cez tieto porty. Paket môže byť preposlaný z jedného prepínača na druhý jedine cez výstupný OpenFlow port na prvom prepínači a vstupný OpenFlow port na druhom prepínači.

OpenFlow prepínač má niekoľko OpenFlow portov prístupných pre OpenFlow operácie. Sada OpenFlow portov nemusí byť identická so sadou sietových rozhraní poskytnutých hardvérom prepínača. Niektoré sietové rozhrania môžu byť vypnuté pre OpenFlow a prepínač môže aj definovať dodatočné OpenFlow porty.

OpenFlow pakety sú prijaté cez **ingress port** (vstupný port) a spracované OpenFlow zreťazeným spracovaním, ktoré ich môže preposlať na **output port** (výstupný port). Vstupný port môže byť použitý pri porovnávaní paketov. OpenFlow zreťazené spracovanie sa môže rozhodnúť poslať paket na výstupný port použitím výstupnej akcie, ktorá definuje ako sa má

paket vrátiť do siete.

OpenFlow prepínač musí podporovať tri typy OpenFlow portov: *physical ports* (fyzické porty), *logical ports* (logické porty) a *reserved ports* (rezervované porty).

2.3.1 Štandardné porty

OpenFlow štandardné porty sú definované ako fyzické porty, logické porty a rezervovaný LOCAL port, ak je podporovaný.

Štandardné porty môžu byť použité ako vstupné a výstupné porty, môžu byť použité v skupinách, majú počítadlo portov a majú svoj stav a konfiguráciu.

2.3.2 Fyzické porty

OpenFlow fyzické porty sú prepínačom definované porty, ktoré korešpondujú hardvérovým rozhraniam prepínača.

V niektorých nasadeniach môže byť OpenFlow prepínač virtualizovaný nad hardvérom prepínača. V takýchto prípadoch môže OpenFlow fyzický port reprezentovať virtuálny diel korešpondujúceho hardvérového rozhrania prepínača.

2.3.3 Logické porty

OpenFlow logické porty sú prepínačom definované porty, ktoré nekorešpondujú priamo s hardvérovým rozhraním prepínača. Logické porty sú vysoko-úrovňové abstrakcie, ktoré môžu byť v prepínači definované použitím non-OpenFlow metód (napr. link aggregation skupiny, tunely, loopback rozhrania).

Logické porty môžu zahŕňať enkapsuláciu paketu a môžu byť mapované na rôzne fyzické porty. Spracovanie vykonané logickým portom je závislé od implementácie a musí byť transparentné pre OpenFlow spracovanie. Interakcia medzi takýmito portami a OpenFlow spracovaním musí prebiehať ako u fyzických portov.

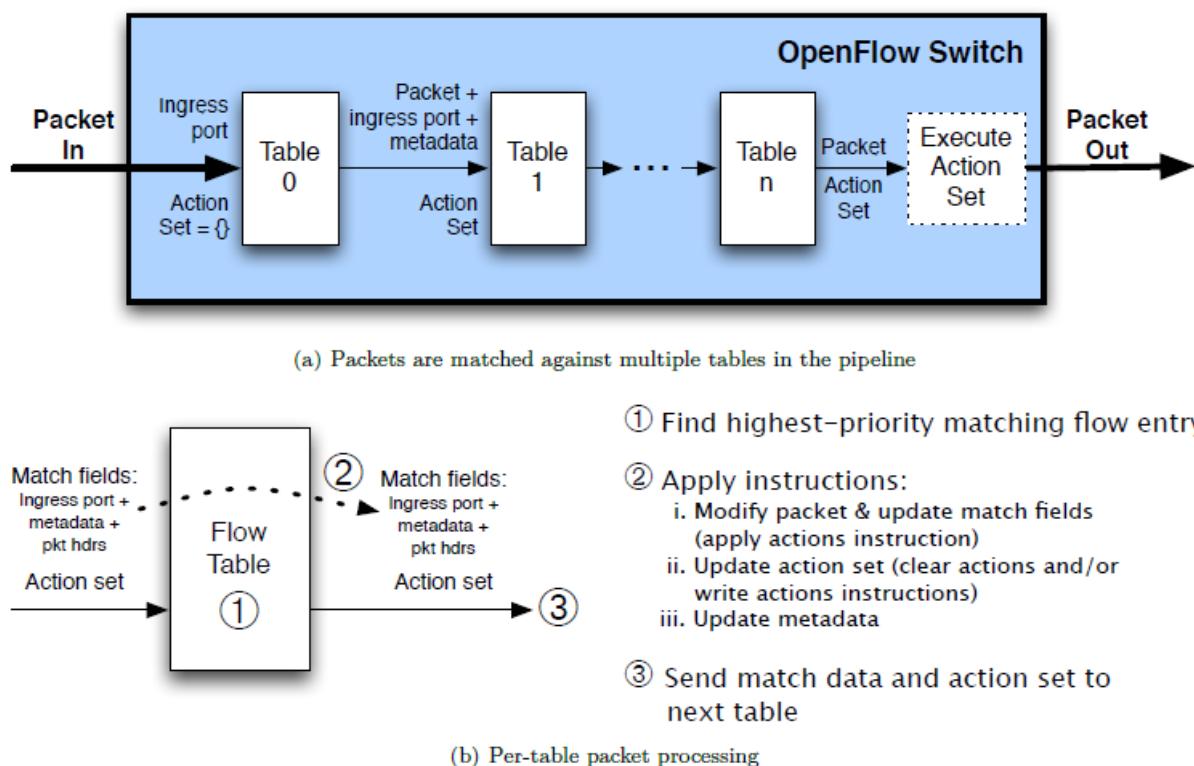
Jediný rozdiel medzi fyzickým a logickým portom je, že paket asociovaný s logickým portom má navyše jedno pole zreteženého spracovania nazývané *Tunnel-ID*. Keď sa paket prijatý na logickom porte odošle OpenFlow kontroléru, jeho logický port aj fyzický port, s ktorým je asociovaný sú kontroléru oznamené.

2.3.4 Rezervované porty

OpenFlow Rezervované porty špecifikujú generické prenosenie akcie, ako posielanie na kontrolér, flooding, alebo prenosenie použitím non-OpenFlow metód, ako tradičné spracovanie prepínačom.

2.4 OpenFlow tabuľky

Táto časť popisuje komponenty tabuľiek tokov a skupinových tabuľiek technikou zhody a akcie.



Obr. 5: Tok paketov cez zretežené spracovanie [5, strana 15]

2.4.1 Zretežené spracovanie

Existujú dva typy prepínačov kompatibilných s OpenFlow protokolom:

- OpenFlow-only - podporuje len OpenFlow operácie. V takomto prepínači sú všetky paky spracované OpenFlow zreteženým spracovaním a nijako ináč.
- OpenFlow-hybrid - podporuje aj OpenFlow operácie, aj bežné operácie Ethernetového prepínača, teda tradičné L2 prepínanie, VLAN, L3 smerovanie, ACL a QoS.

OpenFlow zretežené spracovanie každého OpenFlow logického prepínača obsahuje jeden, alebo viac tabuľiek tokov, pričom každá tabuľka tokov pozostáva z viacerých položiek to-

ku. Toto zretežené spracovanie definuje interakciu medzi paketmi a tabuľkami tokov. OpenFlow prepínač musí mať aspoň jednu tabuľku tokov, pričom voliteľne ich môže byť viac.

Tabuľky tokov v OpenFlow prepínači sú sekvenčne číslované, začínajúc číslom 0. Zretežené spracovanie vždy začína na prvej tabuľke tokov, tzv. paket je najprv porovnávaný s položkami toku v tabuľke tokov 0. Ostatné tabuľky tokov môžu byť použité v závislosti od výsledku porovnávania v prvej tabuľke.

Pri spracovaní tabuľkou tokov, paket je porovnaný s položkami toku tejto tabuľky pre nájdenie zhody. Ak je nájdená zhodná položka toku, vykoná sa jej inštrukčná sada. Tieto inštrukcie môžu explicitne nasmerovať paket do ďalšej tabuľky tokov, kde sa opakuje rovnaký proces. Položka toku môže nasmerovať paket len na tabuľku s väčším číslom ako je tá, v ktorej sa paket momentálne nachádza. Toto spracovanie je teda jednosmerné. Ak zhodná položka toku nenasmeruje paket na ďalšiu tabuľku, zretežené spracovanie tu končí, paket je spracovaný s pridruženou sadou akcií a zvyčajne preposlaný.

2.4.2 Tabuľka tokov

Tabuľka tokov pozostáva z položiek toku.

Match Fields	Priority	Counters	Instructions	Timeouts	Cookie	Flags
--------------	----------	----------	--------------	----------	--------	-------

Tabuľka 1: Tabuľka tokov

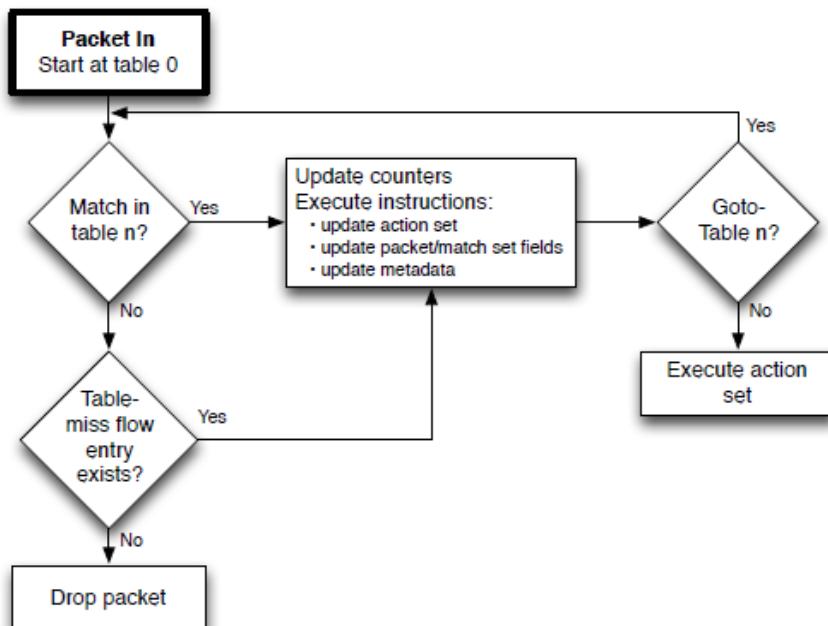
Každá položka tabuľky obsahuje:

- **match fields** - polia na porovnávanie s paketom. Tie pozostávajú z vstupného portu a hlavičiek paketu, prípadne aj iných polí, ako sú metadáta špecifikované predchádzajúcou tabuľkou.
- **priority** - udáva prioritu položky toku.
- **counters** - počítadlá upravované keď sa nájde zhoda s paketom.
- **instructions** - inštrukcie na modifikovanie akčnej sady alebo zreteženého spracovania.
- **timeouts** - maximálne množstvo času alebo čas nečinnosti než tok v prepínači expiruje.
- **cookie** - nejasná dátová hodnota zvolená kontrolérom. Môže ním byť použitá na filtrovanie položiek toku ovplyvnených štatistikami tokov, modifikáciami, alebo požiadavkami na vymazanie tokov.
- **flags** - pozmeňujú spôsob správy položiek toku.

Položka tabuľky tokov je identifikovaná jej poľom *match fields* a *priority*, ktoré spoločne vytvárajú unikátny identifikátor položky pre konkrétnu tabuľku.

Inštrukcia položky toku môže obsahovať akcie, ktoré majú byť vykonané na pakete v niektorom bode zrešteného spracovania.

2.4.3 Porovnávanie



Obr. 6: Diagram popisujúci prechod paketu cez prepínač [5, strana 17]

Pri prijatí paketu, OpenFlow prepínač vykoná funkcie ukázané na diagrame zobrazenom hore. Prepínač začne vyhľadávaním v prvej tabuľke tokov a v závislosti od zrešteného spracovania môže pokračovať vo vyhľadávaní aj v ďalších tabuľkách.

Políčka pre nájdenie zhody sú extrahované z paketu. Ktoré políčka budú pre tento účel vybrané záleží od typu paketu. Typicky v sebe zahŕňajú rôzne políčka hlavičky paketu, ako Ethernetová zdrojová adresa, alebo cieľová adresa protokolu Internet Protocol (IP), zvyčajne IPv4. Porovnávania môžu byť tiež vykonané v závislosti od vstupného portu, od metadát a iných polí. Ak má políčko položky toku hodnotu ANY, zhoda nastane pri všetkých možných hodnotách hlavičky paketu.

Paket je porovnávaný s tabuľkou a len zhodná položka s najvyššou prioritou je vybraná. Počítadlá asociované s vybranou položkou sú upravené a aplikuje sa inštrukčná sada danej položky.

2.4.4 Odstránenie toku

Položky toku sú z tabuľky tokov vymazané v dvoch prípadoch. Bud' na požiadavku OpenFlow kontroléra alebo cez mechanizmus prepínača pre ukončenia platnosti.

Tento mechanizmus je založený na stave a konfigurácii položiek toku. Každá položka toku má s ňou asociované tzv. *idle_timeout* a *hard_timeout*. Ak pole *hard_timeout* je nenulové, prepínač musí dbať na prichodzí čas položky toku, pretože ju môže byť potrebné neskôr zmazať. S nenulovou hodnotou toto pole zapríčinuje odstránenie položky toku za daný počet sekund, pričom nezáleží koľko paketov s ňou našlo zhodu. Ak je pole *idle_timeout* nenulové, prepínač musí dbať na prichodzí čas posledného paketu asociovaného s tokom, pretože ho môže byť neskôr potrebné zmazať. Nenulová hodnota tohto poľa spôsobí zmazanie položky toku ak ku nej nebola nájdená zhoda s paketmi v danom počte sekund.

OpenFlow kontrolér môže aktívne mazať položky toku z tabuľiek tokov posielaním na to určených správ. Ak je však port prepínača pridaný, upravený alebo vymazaný, nezmení to ani neupraví položky toku. Kontrolér ich musí explicitne zmazať alebo upraviť ak je to potrebné.

2.4.5 Tabuľka skupín

Tabuľka skupín pozostáva z položiek skupiny. Schopnosť položky toku ukazovať na skupinu umožňuje protokolu poskytovať ďalšie metódy preposielania. Každá položka skupiny má

Group Identifier	Group Type	Counters	Action Buckets
------------------	------------	----------	----------------

Tabuľka 2: Tabuľka skupín

svoj identifikátor a obsahuje:

- **group identifier** - pole identifikujúce skupinu v rámci OpenFlow prepínača.
- **group type** - stanovuje sémantiky skupín.
- **counters** - upravované pri spracovaní paketov skupinou.
- **action buckets** - zoradený zoznam tzv. ‘vedier akcií’, kde každé takéto vedro pozostáva zo sady akcií pre vykonanie a pridružené parametre.

2.5 OpenFlow komunikácia

OpenFlow kanál je rozhranie spájajúce každý OpenFlow logický prepínač s OpenFlow kontrolérom. Cez tohto rozhranie kontrolér konfiguruje a spravuje prepínač, prijíma oznámenia od prepínača a posiela pakety z prepínača. Kontrolér typicky spravuje OpenFlow prepínače

na diaľku cez viacero sietí. Pre OpenFlow kanály môže byť použitá oddelená sieť (out-of-band spojenie), alebo môžu kanály prúdiť sietou spravovanou OpenFlow prepínačmi (in-band spojenie). Kontrolný kanál prepínača môže podporovať jeden kanál s jedným kontrolérom, alebo aj viacerom kanálov umožňujúcich viacerým kontrolérom zdieľať správu prepínača.

Prepínač podporuje tri typy správ: kontrolér-prepínač, asynchrónne a symetrické.

2.5.1 Správy kontrolér-prepínač

Tieto správy sú zahájené kontrolérom a môžu, ale nemusia, vyžadovať odpoved' od prepínača.

Features: kontrolér si môže vyžiadať identitu a základné schopnosti prepínača poslaním požiadavku vlastností. Prepínač musí odpovedať špecifikovaním svojej identity a svojich základných schopností. Tento proces zvyčajne prebieha pri zakladaní OpenFlow kanála.

Configuration: kontrolér dokáže nastaviť a opýtať sa na konfiguračné parametre prepínača. Ten odpovedá len na dotazy kontroléra.

Modify-State: správy sú kontrolérom posielané pre spravovanie stavu prepínača. Ich hlavný účel je pridávať, mazať a upravovať položky toku/skupiny v OpenFlow tabuľkách prepínača a nastavovať vlastnosti jeho portov.

Read-State: slúžia kontroléru na zber rôznych informácií z prepínača, ako súčasná konfigurácia, štatistiky a vlastnosti.

Packet-out: tieto správy slúžia kontroléru na vyslanie paketov určeným portom prepínača a na preposlanie paketov prijatých cez správy **Packet-in**.

Barrier: tieto správy sú kontrolérom používané pre zaistenie správneho poradia spracovania správ a na prijatie oznamov o splnených operáciach.

Role-Request: kontrolér ich používa na nastavenie role jeho kanála, prípadne jej zistenie. Toto je hlavne užitočné, keď je prepínač spojený z viacerými kontrolérmi.

Asynchronous-Configuration: správy sú používané kontrolérom na nastavenie alebo zistenie stavu dodatočných filtrov použitých pri prijímaní asynchronných správ. Je to bežne využívané pri zakladaní kanála a pri komunikácii prepínača s viacerými kontrolérmi.

2.5.2 Asynchrónne správy

Asynchrónne správy sú posielané prepínačom bez ich vyžiadania kontrolérom. Prepínač ich kontroléru posiela na indikovanie príchodu paketu, zmeny stavu prepínača, alebo chyby.

Packet-in: predáva kontrolu nad paketom kontroléru. Tieto správy sa zvyčajne používajú, keď sa nenájde zhoda paketu s tabuľkami toku a paket sa tak vyšle na kontrolér. Väčšinou sa v správe posiela len časť hlavičky paketu a ID zásobníka, kde je celý paket na prepínači dočasne uložený. ID zásobníka sa použije na identifikovanie paketu, keď dá kontroler prepínaču povel na odoslanie paketu.

Flow-Removed: informuje kontrolér o odstránení položky toku z tabuľky tokov. Sú generované ako výsledok požiadavky kontroléra o vymazanie toku, alebo pri expirovaní toku v prepínači kvôli uplynutiu niektorého z časovačov.

Port-status: informuje kontrolér o zmene na porte. Od prepínača sa očakáva poslanie tejto správy kontroléru pri zmene konfigurácie alebo stavu portu.

Error: prepínač dokáže upozorniť kontrolér na problémy pomocou chybových správ.

2.5.3 Symetrické správy

Symetrické správy sú posielané bez vyžiadania z ľubovoľnej strany.

Hello: správy posielané medzi prepínačom a kontrolérom pri štarte.

Echo: požiadavka môže byť poslaná z obidvoch strán, pričom sa musí vrátiť odpoveď. Používajú sa hlavne pre overenie životnosti spojenia medzi kontrolérom a prepínačom a môžu sa takisto použiť na meranie latencie a šírky pásma.

Experimenter: tieto správy poskytujú štandardný spôsob na poskytnutie rozšírenej funkcionality OpenFlow prepínača v rámci typového priestoru správ. Táto oblasť je určená pre prvky mienené pre budúce revízie protokolu.

2.6 Verzie protokolu OpenFlow

Medzi niektoré z hlavných funkcií, ktoré sa do protokolu od verzie 1.0 pridali patria [10]:

OpenFlow 1.1:

- Viaceré tabuľky - prináša flexibilnejšie zretazenie s viacerými tabuľkami, pre zlepšenie efektívnosti protokolu v zariadeniach.
- Skupiny - tabuľka skupín umožňuje protokolu označovať sadu portov jednej entity pre účely preposielania paketov.
- MPLS a VLAN značky - zlepšená podpora VLAN oproti starším verziám a podpora MPLS.
- Virtuálne porty - podpora virtuálnych portov pre komplexné preposielacie funkcie.

- Zlyhanie spojenia s kontrolérom - ruší starý spôsob a pridáva dva jednoduchšie módy pre riešenie straty spojenia s kontrolérom.

OpenFlow 1.2:

- Podpora pre rozšíriteľné hlavičky - protokol začal používať štruktúru Type-Length-Value (TLV) v poliach *match fields*, *set_field* a *packet_in*, čo dramaticky zvýšilo ich flexibilitu.
- IPv6 - pridaná základná podpora pre porovnávanie a prepisovanie IPv6 hlavičiek.

OpenFlow 1.3:

- Zlepšenie podpory pre IPv6 - pridáva schopnosť porovnávať IPv6 rozšíriteľné hlavičky.
- Tunelovanie - pridanie *Tunnel-ID* metadát umožňujúcich použitie tunelovacích techník.
- Meranie jednotlivých tokov - podpora pre meranie a kontrolu množstva paketov jednotlivých tokov. Jedno z hlavných využití má pri obmedzení množstva posielaných paketov na kontrolér.
- Provider Backbone Bridging (PBB) - pridáva podporu pre označovanie paketov pomocou PBB enkapsulácie. Umožňuje podporu viacerých sietí založených na PBB.

OpenFlow 1.4:

- Zväčšená flexibilita protokolu - veľká časť protokolu už používa TLV štruktúry.
- Optické porty - nová sada vlastností portov pridáva podporu pre optické porty.
- Sledovanie tokov - definuje schému, kde viaceré kontroléry môžu spravovať prepínač. Umožňuje kontroléru sledovať v reálnom čase zmeny na ľubovoľnej podmnožine tabuľky tokov prepínača vykonané inými kontrolérmi.
- Vystahovanie - väčšina tabuľiek tokov má ohraničenú kapacitu. V predošlých verziách pri plnej kapacite neboli nové toky do tabuľiek vkladané a kontroléru bola vracaná chyba, čo následne mohlo narušiť službu. Vystahovanie umožňuje prepínačom automaticky odstrániť položky z nižšou dôležitosťou, pre urobenie miesta novým položkám. Toto napomáha zjednatiť degradáciu správania sa pri zaplnenej tabuľke.
- Udalosť neobsadenosti - rieši rovnaký problém ako v predošлом bode. Táto udalosť pridáva mechanizmus, ktorým dokáže kontroléru poslať skoré varovanie o blížiacom sa kapacitnom limite. Prahovú hodnotu pre zaslanie varovania nastavuje kontrolér.
- Zväzky - pridáva mechanizmus zväzkov, umožňujúci aplikovať skupinu OpenFlow správ ako jednu operáciu. Toto napomáha lepšej synchronizácii zmien naprieč prepínačmi.

OpenFlow 1.5:

- Výstupné tabuľky - v predchádzajúcich verziách špecifikácie boli všetky výpočty robene v kontexte vstupného portu. Táto verzia prináša výstupné tabuľky umožňujúce výpočty v rámci výstupného portu. Keď je paket poslaný na výstupný port, začne sa jeho spracovanie na prvej výstupnej tabuľke, kde môžu položky toku určiť ďalšie akcie na vykonanie.
- Zrečazené spracovanie rozpoznávajúce typ paketu - v predchádzajúcich verziách protokolu museli byť všetky pakety Ethernetové. Od verzie 1.5 je protokol schopný rozpoznávať typy paketov, čo umožňuje pracovať s inými typmi, ako sú IP pakety alebo Point-to-Point Protocol (PPP) pakety.
- Rozšíriteľné štatistiky položiek toku - v predošlých verziách sa používali fixné štruktúry pre štatistiky položiek toku. Od verzie 1.5 je zavedené flexibilné zakódovanie, umožňujúce zakódovať ľubovoľné umelé štatistiky položiek toku.
- Spúštač štatistik položiek toku - dopytovanie sa po štatistikách položiek toku môže vyvoláť vysokú dátovú réžiu a vyťaženie prepínača. Nový spúšťiaci mechanizmus pre štatistiky umožní ich automatické zasielanie na kontrolér na základe rôznych prahových hodnôt štatistik.
- Stav spojenia s kontrolérom - umožňuje kontroléru zistiť stav všetkých spojení z prepínača na kontroléry. To umožňuje kontroléru detektovať rozdelenie riadiacej siete alebo monitorovať stav ostatných kontrolérov.

3 Ciele práce

Cieľom mojej diplomovej práce bolo uviesť problematiku SDN sietí a pilotne zrealizovať zapojenie takejto siete za pomoci protokolu OpenFlow, kde som mal ukázať jej architektúru, používané protokoly a princípy fungovania takejto siete. Siet som mal realizovať vo virtuálnej podobe, prostredníctvom emulátora Mininet a vo fyzickej podobe, za pomoci vstavaného počítača v úlohe OpenFlow prepínača.

V časti zameranej na analýzu budú preskúmané a vybrané riešenia pre jednotlivé komponenty SDN siete, ktoré budú použité pri praktickej ukážke. Medzi tieto komponenty sa radí OpenFlow prepínač a SDN kontrolér.

V rámci fyzického prevedenia OpenFlow prepínača sa vykoná prieskum a voľba jedného z dostupných vstavaných počítačov a preskúmané budú aj alternatívy, ako NetFPGA, programovateľné procesory alebo využitie OpenFlow protokolu na plnohodnotných fyzických prepínačoch.

Analyzované bude aj softvérové vybavenie jednotlivých SDN komponentov. Preskúmané budú riešenia pre OpenFlow prepínač, realizovaného prostredníctvom vstavaného počítača ako aj virtuálne cez emulátor Mininet. Rovnako sa vykoná prieskum a výber SDN kontroléra, ktorý bude spustený na počítači/servery.

V praktickej časti budú rozobraté jednotlivé ukážkové príklady zapojenia infraštruktúry pre SDN sieti a ich pripojenie na SDN kontrolér. Bude ukázané nastavenie SDN kontroléra, spôsob fungovania vytvorenej siete a možnosti jej správy prostredníctvom SDN kontroléra.

Posledná časť sa bude zaoberať možnosťami rozšírenia SDN kontroléra o nové funkcie ako aj spôsob vývoja aplikácií pre správu SDN kontroléra.

V závere budú načrtnuté možnosti výučby SDN sietí na katedre KIS a smer, ktorým by sa mohlo ďalšie skúmanie týchto sietí na KIS uberať.

4 Analýza a výber SDN komponentov

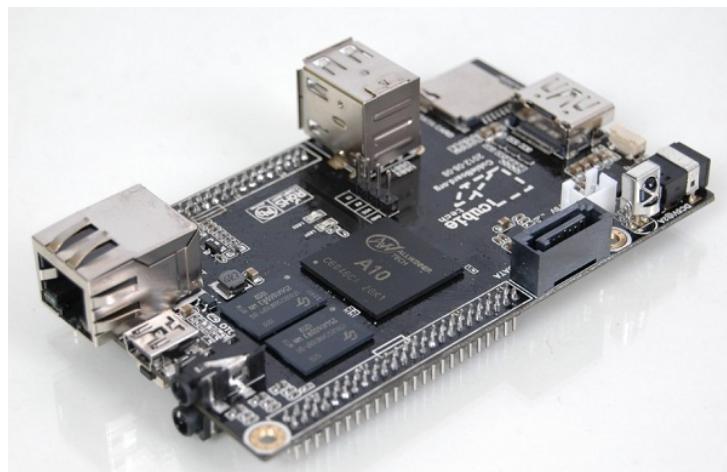
V tejto časti bude vykonaná analýza a výber jednotlivých komponentov SDN riešenia:

- OpenFlow prepínač - realizovaný vo forme vstavaného počítača, ako zariadenia vhodného pre rôzne druhy testovania. Bude preskúmaná ponuka dostupných a cenovo prijateľných modelov vstavaných počítačov vhodných pre úlohu prepínača ako aj dostupné operačné systémy a softvérové riešenie OpenFlow prepínača, ktoré v zariadení použijem. Takisto budú preskúmané aj iné možnosti realizácie platformy OpenFlow prepínača, ako sú NetFPGA, programovateľné sieťové procesory a hardvérové prepínače.
- SDN kontrolér - realizovaný vo forme počítača s vybraným softvérom SDN kontroléra. Preskúmané budú dostupné softvérové riešenia SDN kontroléra.

4.1 Analýza vstavaných počítačov

4.1.1 Cubieboard1

Cubieboard1 je malá, technikom priateľská, rozšíriteľná, lacná a výkonná ARM doska s osadením Allwinner A10 SoC [11].



Obr. 7: Cubieboard1 [12]

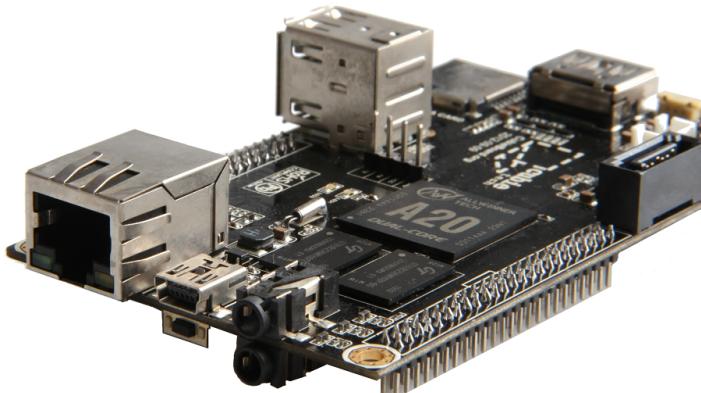
Hardvérová špecifikácia:

- AllWinnerTech SoC A10; ARM Cortex-A8 (1GHz), ARM Mali400 MP1 GPU.
- 1GB DDR3 RAM.
- 4GB vnútorná NAND flash, microSD slot, do 2TB na 2.5 SATA disk.
- 5VDC vstup 2A alebo USB otg vstup.
- 1x 10/100 Ethernet, podpora usb wifi.

- 2x USB 2.0 HOST, 1x micro USB 2.0 OTG.
- 1x HDMI 1080P obrazový výstup.

4.1.2 Cubieboard2

Cubieboard2 je novšia verzia dosky Cubieboard1. Je s ňou kompatibilná a zdieľajú rovnakú dosku plošných spojov, ale disponuje dvojjadrovým CPU a GPU [11].



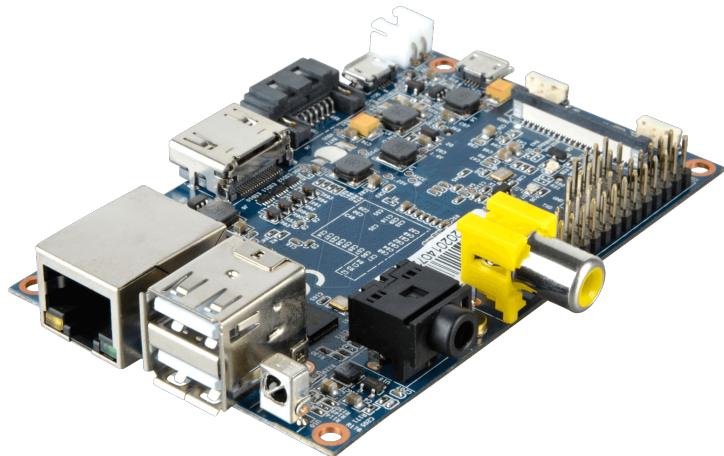
Obr. 8: Cubieboard2 [13]

Hardvérová špecifikácia:

- AllWinnerTech SoC A20; ARM Cortex-A7 Dual-Core (1GHz), ARM Mali400 MP2 GPU.
- 1GB DDR3 RAM.
- 4GB vnútorná NAND flash, microSD slot, do 2TB na 2.5 SATA disk.
- 5VDC vstup 2A alebo USB otg vstup.
- 1x 10/100 Ethernet, podpora usb wifi.
- 2x USB 2.0 HOST, 1x micro USB 2.0 OTG.
- 1x HDMI 1080P obrazový výstup.

4.1.3 Banana Pi

Banana Pi je cenovo dostupný jednodoskový počítač. Je univerzálny a má široké možnosti konfigurácie. Má osadenie Allwinner A20 SoC [14].



Obr. 9: Banana Pi [15]

Hardvérová špecifikácia:

- AllWinnerTech SoC A20; ARM Cortex-A7 Dual-Core (1GHz), ARM Mali400 MP2 GPU.
- 1GB DDR3 RAM.
- SD slot, do 2TB na 2.5 SATA disk.
- 5VDC vstup 2A alebo USB otg vstup.
- 1x 10/100/1000 Ethernet, podpora usb wifi.
- 2x USB 2.0 HOST, 1x micro USB 2.0 OTG.
- 1x HDMI 1080P obrazový výstup.

4.1.4 Raspberry Pi model B+

Raspberry Pi je jednodoskový počítač vyvinutý nadáciou Raspberry Pi Foundation pre účely výučby základov počítačových vied v školskom prostredí. [16].



Obr. 10: Raspberry Pi model B+ [17]

Hardvérová špecifikácia:

- Broadcom BCM2835; ARMv6 SoC (700MHz), Broadcom VideoCore IV.
- 512MB SDRAM.
- microSD slot.
- 5VDC vstup 1.2A alebo USB otg vstup.
- 1x 10/100 Ethernet, podpora usb wifi.
- 4x USB 2.0 HOST, 1x micro USB 2.0 OTG.
- 1x HDMI 1080P obrazový výstup.

4.1.5 Vybraný produkt

Uvedené boli porovnané na základe ich ceny a vlastností.

názov	Cubieboard1	Cubieboard2	Banana Pi	Raspberry Pi B+
orientačná cena	48 eur	64 eur	52 eur	40 eur
CPU (ARM)	v7 1GHz	v7 1GHz Dual-Core	v7 1GHz Dual-Core	v6 700MHz
GPU (ARM)	Mali400 MP1	Mali400 MP2	Mali400 MP2	VideoCore IV
pamäť RAM	1GB DDR3	1GB DDR3	1GB DDR3	512MB SDRAM
vstavané úložisko	4GB NAND	4GB NAND	-	-
pamäťový slot	microSD	microSD	SD	microSD
USB 2.0	2x	2x	2x	4x
Ethernet	100Mbit/s	100Mbit/s	1000Mbit/s	100Mbit/s

Tabuľka 3: Porovnanie vstavaných počítačov

Z porovnávaných zariadení bol vybraný počítač Banana Pi, ktorý disponuje výkonným procesorom, dostatočne veľkou pamäťou RAM a Gigabitovým Ethernet portom. Jeho kúpa je rovnako výhodná vzhľadom na pomer výkonu a ceny.

Kedže jeden Ethernetový port, ktorý sa na zariadení nachádza je nedostatočný pre vykonávanie funkcie fyzického prepínača, ku zariadeniu boli dokúpené aj dva USB-to-Ethernet adaptéry, ktoré sa zapoja do dvoch USB 2.0 portov na zariadení a urobia z nich 100Mbps Ethernetové porty. Zvolené boli adaptéry *i-tec USB 2.0 Fast Ethernet Adapter 100/10Mbps* obsahujúce chipset Asix AX88772B. Tento chipset obsahuje vysoko výkonný a efektívny ASIC, vďaka ktorého nízkej spotrebe nie je nutné zaobstaráť externé napájanie pre adaptéry.

Architektúra procesora v tomto zariadení dovoľuje spúštať aj známe distribúcie Linuxu, ako sú napríklad Ubuntu, Debian, OpenSuse, Fedora, alebo aj Android. Dostupné sú samoz-

rejme aj distribúcie upravené na rôzne druhy vstavaných počítačov ako Raspbian, Cubian alebo odľahčené Lubuntu. Pre Banana Pi je však urobená vlastná distribúcia s názvom Bananian Linux, čo je odľahčená a prispôsobená verzia OS Debian 7, ktorá bude vhodnou voľbou pre implementáciu prepínača.

4.2 NetFPGA a programovateľné sietové procesory

4.2.1 NetFPGA

Platforma NetFPGA umožňuje výskumníkom a inštruktorom stavať fungujúce prototypy vysokorychlostných, hardvérovo akcelerovaných sietových systémov [18]. Využíva sa v školskom prostredí pre výučbu tvorenia Ethernetových prepínačov a IP smerovačov použitím skôr hardvéru, než softvéru. Používa sa tiež výskumníkmi pre testovanie pokročilých služieb pre siete novej generácie.

Je to otvorená platforma dostupná vývojárom celosvetovo. Referenčné návrhy pre tento systém obsahujú IPv4 smerovač, Ethernetový prepínač a sietovú kartu so štyrmi portami. Pozostáva z programovateľnej vývojárskej dosky, referenčných implementácií a vzorky výukových programov. Samotná doska je kartou Peripheral Component Interconnect (PCI) a môže byť vložená do ľubovoľného počítača s príslušným slotom. Na doske sa nachádza používateľom programovateľné Field Programmable Gate Array (FPGA), statická a dynamická Random-access Memory (RAM) a štyri Ethernet porty o rýchlosťi 1Gbps.

Do hardvéru tejto karty je možné naprogramovať jednoduché správanie. V takom prípade môže karta pracovať aj samostatne, bez potreby počítača. Pri zložitejších programoch si ale vyžaduje prepojenie s počítačom, na ktorom sa bude v softvéri nachádzať hlavná logika programu. To je aj prípad u protokolu OpenFlow.

Karta by dokázala v kombinácii s počítačom zastávať úlohu OpenFlow prepínača. Mala by v sebe naprogramovanú základnú logiku OpenFlow prepínača a bola by pripojená na počítač, v ktorom sa nachádza OpenFlow klient starajúci sa o komplexné operácie protokolu. Takýmto riešením by sa mal dosiahnuť OpenFlow prepínač schopný preposielať pakety rýchlosťou linky.

4.2.2 Štandardný prepínací hardvér

Pri súčasnej tvorbe sietových zariadení, ako sú smerovače alebo prepínače, je len pár komponentov špeciálne navrhnutých pre svoj účel, pričom zvyšok pozostáva zo štandardných,

masovo vyrábaných komponentov. Veľa sietových spoločností prešlo v posledných rokoch na používanie štandardných matičných dosiek od Intelu. Urobili tak po pokrokoach firmy Intel a ich x86 procesorov [19].

Zároveň ako začínajú predajcovia propagovať architektúru sietí, ktorá oddeluje softvér od hardvéru, štandardný prepínací hardvér sa presúva do popredia pozornosti. Tieto prepínače sú postavené na veľkoobchodných štandardných výrobných modeloch a sú predávané bez OS. Vyrábajú ich spoločnosti špecializujúce sa na návrh a výrobu kremíkových čipov. Tento hardvér je zvyčajne postavený na x86 architektúre procesorov a čipových súpravách od výrobcov, ako napríklad Broadcom, Intel, Mellanox a Marvell. V súčasnosti môžu ešte poskytovať menej funkcia ako prepínače s proprietárnymi čipmi, ale sú lacnejšou a flexibilnejšou alternatívou. Ako motiváciu pre kúpu takýchto prepínačov sa udáva [20]:

- Vyhnutie sa závislosti od predajcu sietových zariadení - voľnosť výberu a zmeny predajcu hardvérového alebo softvérového sietového vybavenia.
- Zniženie operačných nákladov vybudovaním siete splňajúcej potreby používateľa - výber spôsobu správy siete s ktorou má používateľ už znalosti, namiesto vynakladania financií na tréningy zamestnancov, požiadavky na ich znalosti a pod. kvôli obmedzeným možnostiam správy v tradičných sietových zariadeniach.
- zníženie nákladov na kúpu zariadení vďaka transparentnejšej konkurencii - ľahšie porovnanie generických zariadení umožní lepšiu orientáciu v pomere výkon/cena zariadení.

Veľkú úlohu v rozšírení záujmu o tieto prepínače hralo vytvorenie prostredia Open Network Install Environment (ONIE). Je to malý OS, nachádzajúci sa v týchto štandardných prepínačoch ako firmware, ktorý zjednodušuje inštaláciu plnohodnotných sietových OS do prepínača [21].

Tento spôsob osobitného zaobstarávania prepínacieho hardvéru a softvéru je ale stále pomerne nový a málo preskúmaný. Preto veľa spoločností ostáva u tradičného modelu a predajcov, ako napríklad Cisco, ktorý ponúkajú stabilné riešenia a kvalitný servis.

Donedávna si takýto štandardný hardvér kupovali len veľké spoločnosti, ako napríklad Google alebo Facebook. Vývinom trhu a príchodom technológií, ako ONIE ,sa ale začína o tento produkt zaujímať viac spoločností. Firma Dell bola prvou z veľkých firiem, ktorá začala tieto produkty verejne predávať zákazníkom a v súčasnosti tak urobila aj firma HP. Dnes ponúkajú takýto hardvér spoločnosti ako Dell, HP, Quanta (IW Networks), Accton (Edge-Core

Networks) a Delta Networks (Agema Systems). Spoločnosti Big Switch Networks, Cumulus Networks a Pica8 ponúkajú navyše aj kompletné riešenie s prepínacím hardvérom získaným od svojich partnerov a vlastným sietovým OS.

Medzi dostupné sietové OS pre takéto prepínače patria: SwitchLight OS (Big Switch Network), Cumulus Linux (Cumulus Networks), FastPath (Broadcom), PicOS (Pica8).

4.2.3 Špecializovaný prepínací hardvér

Firma Cisco poskytuje vo svojich zariadeniach podmnožinu funkcií protokolu OpenFlow 1.0 a 1.3 prostredníctvom OpenFlow zásuvného modulu [22]. Spoločnosť má podporu protokolu v produkčných radách zariadení ASR, Nexus a Catalyst. Cisco sa však viac zameriava na svoje vlastné SDN riešenie, používajúce protokol OpFlex, ako alternatívu ku OpenFlow.

Firma Juniper taktiež poskytuje v niektorých svojich zariadeniach podporu pre OpenFlow 1.0 a 1.3 vo forme inštalačného balíčka do Junos OS [23]. Podpora sa vzťahuje na smerovače série MX a prepínače série EX a QFX a to od verzie systému Junos OS 13.2.

4.2.4 Programovateľné sietové procesory

Sietový procesor je integrovaný obvod so sadou vlastností špecificky určených pre sietové aplikácie. Sú to typicky softvérovo programovateľné zariadenia s generickými charakteristikami, podobné univerzálnym procesorom bežne používaným v mnohých druhoch zariadení. Tieto procesory však majú pre zachovanie svojej efektívnosti obmedzenú flexibilnosť.

Spoločnosť EZchip ale napríklad prichádza s novou sadou procesorov nazývaných Network Processors for Smart networks (NPS) [24]. NPS prinášajú maximálnu flexibilnosť cez programovanie založené na jazyku C a úplné 7-vrstvové spracovanie. Tieto vlastnosti umožňujú chod širokého množstva sietových služieb extrémne vysokým výkonom a ktoré dokážu byť vyvolané jednoduchou softvérovou aktualizáciou. NPS zvýrazňuje potrebu novej triedy sietových procesorov, ktoré poskytnú jasné oddelenie riadiacej a dátovej úrovne v novej generácii sietí, kde sú využívané technológie, ako SDN či NFV. EZchip na začiatku uvedie NPS modely s priepustnosťou 200 až 400Gbit/s.

4.3 Analýza softvérových prepínačov

4.3.1 Indigo Virtual Switch

Indigo Virtual Switch (IVS) [25] je voľne dostupný, odľahčený, vysoko výkonný virtuálny prepínač vytvorený s podporou pre protokol OpenFlow. Je kompatibilný s virtuálnou infraštruktúrou Kernel-based Virtual Machine (KVM) a využíva Open vSwitch modul pre preposielanie paketov. IVS je postavený na štruktúre Indigo a využíva generátor kódu LoxiGen na zaobchádzanie s OpenFlow správami. Je navrhnutý pre široko škálovateľnú sieťovú virtualizáciu a podporuje distribúciu naprieč viacerými fyzickými servermi za pomoci SDN kontroléra s podporou OpenFlow. Momentálne podporuje OpenFlow 1.0.

4.3.2 Open vSwitch

Open vSwitch [26] je viacvrstvový softvérový prepínač, vhodný do produkčného prostredia, ktorý je voľne šíriteľný pod licenciou Apache 2. Podporuje štandardné rozhrania pre manažment a povoluje programové rozšírenie a kontrolu preposielacích funkcií.

Je vhodný pre funkciu virtuálneho prepínača vo VM prostrediach. Okrem odhalenia rozhraní štandardnej kontroly a viditeľnosti virtuálnej sieťovej vrstvy, podporuje distribúciu naprieč viacerými fyzickými servermi. Prepínač podporuje viaceré virtualizačné techniky založené na OS Linux, ako Xen/XenServer, KVM a VirtualBox.

Prevažná časť kódu je napísaná v platformovo nezávislom kóde jazyka C a dá sa jednoducho preniesť do iných prostredí. Súčasné vydanie prepínača podporuje nasledovné funkcie:

- Štandard 802.1Q VLAN model s trunk a access portami.
- Párovanie sietových kariet z alebo bez agregácie liniek na nadradenom prepínači.
- NetFlow, sFlow(R) a zrakadlenie pre zvýšenú viditeľnosť.
- Konfiguráciu QoS a politiky.
- Tunelovacie techniky Geneve, Generic Routing Encapsulation (GRE), GRE cez Internet Protocol Security (IPsec), Virtual Extensible LAN (VXLAN) a LISP.
- 802.1ag správu porúch konektivity.
- Podpora OpenFlow po verziu 1.3+ a postupné dopĺňanie funkcií pre vyššie verzie.
- Transakčná konfiguračná databáza s väzbou na jazyky C a Python.
- Vysoko výkonné preposielanie použitím modulu v jadre Linuxu.

4.3.3 Vybraný produkt

IVS o sebe tvrdí, že má otvorennejší kód a je určený čisto na komunikáciu cez OpenFlow čo z neho robí odľahčený a dobre škálovateľný prepínač. Jeho komunita je však malá a neaktívna, keďže ani jeho vývoj sa dlhú dobu neposunul a stále podporuje len počiatočnú verziu 1.0 protokolu OpenFlow.

Open vSwitch je naproti tomu široko uznávaný ako de facto štandardná OpenFlow implementácia. Má veľkú a veľmi aktívnu komunitu a podporuje široké množstvo funkcií, pričom dobre drží krok s vývojom protokolu OpenFlow. Je takisto overený mnohými firmami, ktoré ho používajú v produkčnom prostredí.

Po porovnaní týchto prepínačov bolo vybrané riešenie Open vSwitch.

4.4 Analýza SDN kontrolérov

V tejto časti sú uvedené voľne dostupné SDN kontroléry s otvoreným zdrojovým kódom, na ktorých sa zakladá aj prevažná väčšina kommerčne predávaných SDN kontrolérov rôznych spoločností.

4.4.1 Floodlight

Kontrolér Floodlight [27] je OpenFlow kontrolér napísaný v jazyku Java a vydávaný pod licenciou Apache, ktorý je určený pre nasadenie do podnikov. Je podporovaný otvorenou komunitou vývojárov vrátane niekoľkých inžinierov zo spoločnosti Big Switch Networks a je touto komunitou aktívne testovaný a zlepšovaný. Medzi jeho zmieňované výhody patria:

- Podporuje systém zavádzania modulov pre jednoduché rozširovanie a zlepšovanie.
- Jednoducho nastaviteľný s minimálnymi závislosťami.
- Podporuje široké spektrum virtuálnych a fyzických OpenFlow prepínačov.
- Zvláda správu zmiešaných sietí s OpenFlow a tradičnými zariadeniami - dokáže spravovať ‘ostrovčeky’ fyzických OpenFlow prepínačov.
- Navrhnutý pre vysoký výkon - má jadro z kommerčného produktu spoločnosti Big Switch Networks.
- Podpora platformy OpenStack pre orchesteráciu clouдов.

4.4.2 Ryu

Ryu [28] je aplikačným rámcom pre SDN siete založený na komponentoch a napísaná v jazyku Python, ktorý je šírený pod licenciou Apache 2. Poskytuje softvérové komponenty s dobре definovaným rozhraním, ktoré uľahčuje vývojárom vytváranie nových sieťových riadiacich a kontrolných aplikácií. Ryu podporuje rôzne protokoly pre správu sieťových zariadení, ako OpenFlow, NETCONF, OpenFlow Management and Configuration Protocol (OF-CONFIG) a ďalšie. Kontrolér plne podporuje OpenFlow protokol po verziu 1.4 a rozšírenia firmy Nicira.

4.4.3 ONOS

Open Networking Operating System (ONOS) [29] je prvým voľne šíriteľným SDN systémom zameraným špecificky na ISP a kriticky dôležité siete. ONOS je vytvorený za účelom poskytovania vysokej dostupnosti, škálovateľnosti a výkonu, ktoré si tieto siete vyžadujú. Navyše vytvoril užitočné Northbound abstrakcie a rozhrania na zjednodušenie vývoja aplikácií a Southbound abstrakcie a rozhrania na kontrolu OpenFlow a tradičných zariadení. Prináša:

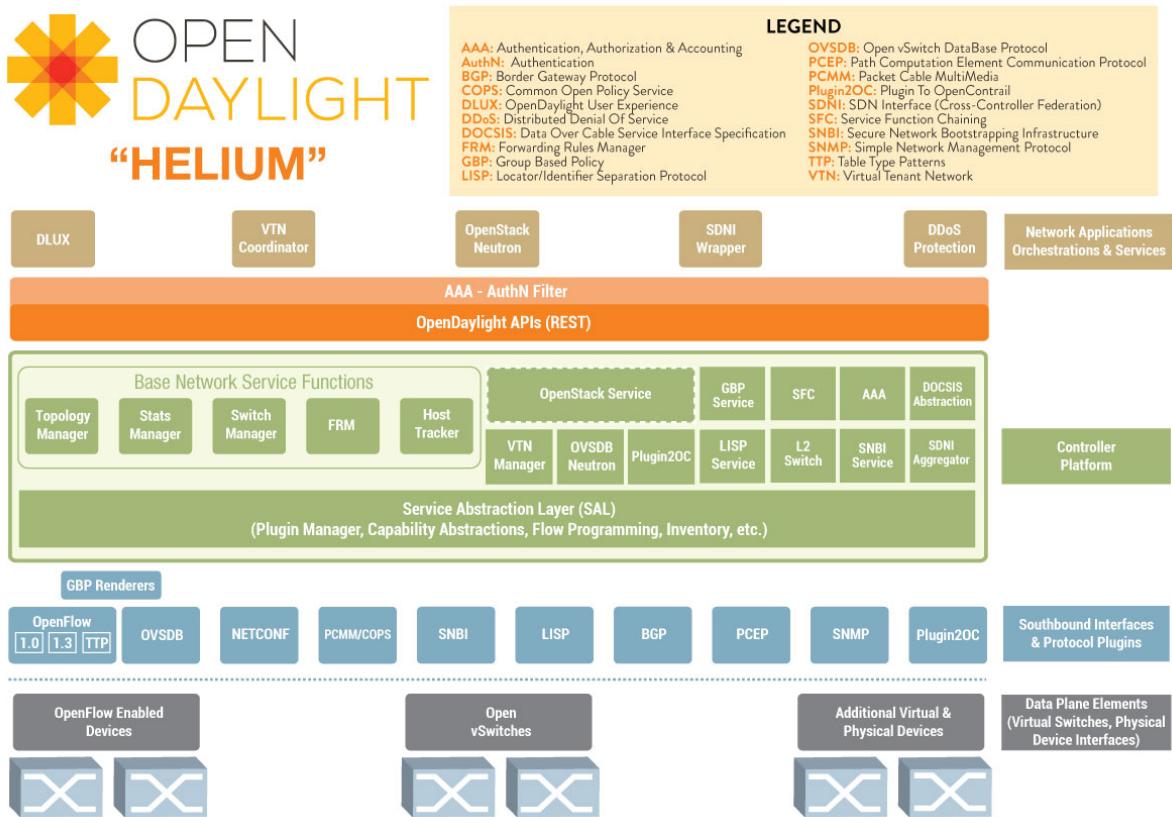
- Vlastnosti prenosových sietí (škálovateľnosť, dostupnosť a výkon) do riadiacej úrovne SDN.
- Agilnosť v štýle webových aplikácií.
- Pomôže ISP s migráciou ich existujúcich sietí na ‘otvorené’ zariadenia.
- Zniženie kapitálových a prevádzkových nákladov ISP.

ONOS bol vyvíjaný v spolupráci s poprednými ISP (AT&T, NTT Communications), predajcami sieťových zariadení (ako Ciena, Ericsson, Fujitsu, Intel) a organizáciou ONF pre validáciu architektúry cez skutočné prípady použitia. Je šírený pod licenciou Apache 2.

4.4.4 OpenDaylight Helium

OpenDaylight [30] je projekt s otvoreným kódom, na ktorom spolupracuje viacero skupín, a ktorý napomáha šíreniu SDN. Je to komunitou vedený, otvorený a priemyslom podporovaný aplikačný rámec. Pozostáva z kódu a plánov pre akceleráciu adopcie SDN, podporovanie ich inovácie, redukciu riziku a vytváranie transparentnejšieho prístupu ku SDN. OpenDaylight je jedným z najväčších úsilí pre spoločný vývoj otvoreného kódu v histórii. Ako taký sa spoľieha na organizáciu Linux Foundation ako hostujúcu entitu pre organizovanie spolupráce v tomto projekte. Softvér tvorený projektom OpenDaylight je napísaný prevažne v jazyku Java a dostupný pod licenciou Eclipse Public Licence (EPL).

OpenDaylight je otvorená platforma pre sietovú programovateľnosť zavádzajúca SDN a NFV do sietí ľubovoľných veľkostí. Druhé vydanie tejto komunity sa nazýva Helium [31] a prichádza s nových používateľským rozhraním a zjednodušeným, upraviteľným inštalačným procesom postaveným na kontajnery Apache Karaf.



Obr. 11: OpenDaylight Helium diagram [31]

Helium pozostáva z viacerých vrstiev, rozdelených podľa ich funkcií:

- **Sietové aplikácie a orchestrácia** - najvyššia vrstva pozostáva z firemných aplikácií a logiky siete, ktoré riadia a monitorujú správanie siete. Komplexnejšie riešenia orchestrácie potrebné pre cloud a NFV spájajú služby dokopy a upravujú sietovú prevádzku podľa potrieb týchto prostredí.
- **Platforma kontroléra** - stredná vrstva je aplikačným rámcem, v ktorom sa prejavujú SDN abstrakcie, poskytujúce množinu spoločných rozhraní do aplikačnej vrstvy (rozhrania Northbound API). Zároveň implementujú jeden alebo viacero protokolov pre riadenie fyzického hardvéru v sieti (rozhrania Southbound API).
- **Fyzické a virtuálne sietové zariadenia** - spodná vrstva pozostáva z fyzických a virtuálnych zariadení, prepínačov, smerovačov a iných prvkov, ktoré vytvárajú spojovací materiál medzi všetkými koncovými bodmi v sieti.

OpenDaylight je voľne šíriteľný projekt s modulárnou, zásuvnou a flexibilnou platformou kontroléra v jeho jadre. Tento kontrolér je implementovaný striktne v softvéri a je obsiahnutý vo svojej vlastnej Java Virtual Machine (JVM). Ako taký môže byť nasadený na ľubovoľnom hardvéri a OS podporujúcom platformu Java.

Kontrolér odhaľuje otvorené rozhrania Northbound API využívané aplikáciami. Podporuje Open Service Gateway initiative (OSGi) aplikačný rámec a obojsmerný REST pre Northbound API. OSGi je používaný pre aplikácie, ktoré budú spustené v rovnakom adresnom priestore ako kontrolér zatiaľ čo REST API (webové rozhranie) je použité pre aplikácie, ktoré nie sú spustené v rovnakom adresnom priestore (alebo na rovnakom stroji) ako kontrolér. Firemná logika a algoritmy spočívajú v aplikáciach. Tie používajú kontrolér pre zbieranie informácií o sieti, beh algoritmov pre vykonanie analýz a následne, ak je to potrebné, použijú kontrolér pre orchestráciu nových pravidiel naprieč sieťou.

Samotný kontrolér obsahuje kolekciu dynamických zásuvných modulov vykonávajúcich potrebné sieťové úlohy. Sú tam série základných sieťových služieb pre úlohy, ako sú zistenie druhu zariadení obsiahnutých v sieti a ich schopnosti, zbieranie štatistik a pod.

Rozhranie Southbound API sú schopné podporovať viacero protokolov v podobe separovaných zásuvných modulov. Tieto moduly sú dynamicky spojené s vrstvou Service Abstraction Layer (SAL). SAL odhaľuje služby zariadenia, na ktoré sú napísané moduly vo vyššej vrstve. SAL určuje ako splniť požadovanú službu bez ohľadu na pod ňou ležiace protokoly používané medzi kontrolérom a sieťovými zariadeniami. To zároveň zaručuje určitú investičnú ochranu pre vytvorené aplikácie, ktoré sa nebudú musieť meniť (alebo nie príliš moc), aj napriek tomu že sa Southbound protokoly časom vyvíjajú.

4.4.5 HP VAN SDN kontrolér

HP VAN SDN kontrolér [32] poskytuje jednotný bod riadenia v OpenFlow sieti. Zjednoduší je správu, obstarávanie a orchestráciu. Umožňuje nástup novej generácie aplikačne založených sieťových služieb a poskytuje otvorené rozhrania API. Tieto rozhrania umožňujú vývojárom tretích strán nasadiť inovačné riešenia pre dynamické prepojenie firemných požiadaviek a sieťovej infraštruktúry cez prispôsobené Java programy alebo univerzálnie RESTful riadiace rozhranie. Tento kontrolér je navrhnutý pre pôsobenie v kampusoch, dátových centrách, alebo v prostredí ISP. Medzi črty kontroléra patria:

- Platforma kontroléra bežiaca v prostredí JVM, postavená na OSGi aplikačnom rámci.

- Podpora pre protokol OpenFlow 1.0 a 1.3.
- Distribučná platforma pre vysokú dostupnosť a škálovateľnosť.
- Vstavané Java aplikácie, rozšíriteľné REST API a Graphical User Interface (GUI).
- Podpora pre zoskupenie troch kontrolérov.

4.4.6 Vybraný produkt

Riešenie ONOS má podporu významných firiem, ale je špecificky zamerané pre problematiku ISP sietí. Ryu a Floodlight majú jednoduchú inštaláciu a spúšťanie. Kontrolér Ryu je však zatial viac určený pre testovacie účely, pretože nie je dostatočne vyvinutý na nasadenie do produkčného prostredia. Floodlight je oproti tomu vyspelejší aj vďaka svojej väzbe na spoločnosť Big Switch Networks, ktorá sa zúčastňuje jeho vývoja. Preto je vhodný aj do produkčného prostredia spoločnosti. Kontrolér firmy HP je navrhnutý na spoluprácu s platformou OpenStack pre poskytnutie uceleného virtuálneho sieťového prostredia. Je to komerčný produkt, takže má stabilnú podporu od firmy HP. Spomedzi týchto platform ale najviac vyniká OpenDaylight Helium. Je to najrozšiahlejšie SDN riešenie, ponúkajúce podporu pre širokú škálu protokolov, pričom sa ale vyhýba prílišnej komplexnosti vďaka svojej štruktúre založenej na zásuvných moduloch. Podielajú sa na ňom skoro všetky významné spoločnosti (viaceré stojace aj za inými SDN kontrolérmi) a je tvorený princípom ‘najlepší kód vyhľadáva’, pričom v sebe zahŕňa najlepšie časti z viacerých SDN kontrolérov. Takisto má najväčšiu komunitu a je na ňom založených niekoľko komerčne dostupných SDN riešení od rôznych spoločností, ako napríklad Cisco, IBM, Brocade a mnoho ďalších.

Po porovnaní týchto kontrolérov bol preto zvolený kontrolér OpenDaylight Helium.

5 Ukážky infraštruktúry SDN siete

Táto časť obsahuje ukážku zapojenia a nastavenia infraštruktúry SDN siete v troch rôznych variantoch. Samotná funkčnosť bude overená v nasledujúcej časti, zaobrájúcej sa SDN kontrolérom. Medzi tri varianty siete patria:

- Virtuálna sieť - emulovaná na jednom počítači v programe Mininet, pozostávajúca z virtuálnych prepínačov, počítačov a liniek medzi týmito zariadeniami.
- Fyzická sieť - tvorená fyzickým prepínačom v podobe vstavaného počítača Banana Pi a dvoma počítačmi, ktoré sú na prepínač pripojené.
- Zmiešaná sieť - tvorená dvoma prepínačmi (Banana Pi a počítač s Open vSwitch), na ktoré je pripojený fyzický počítač a dva VM (obsiahnuté v počítači s Open vSwitch).

Na konci tejto kapitoly je na malom príklade opísaná práca s OpenFlow tokmi v prepínači Open vSwitch.

5.1 Virtuálna sieť

V tejto podkapitole je opísaná práca s emulátorom Mininet, v ktorom bola vytvorená prvá varianta SDN sietovej infraštruktúry, teda sieť s virtuálnymi zariadeniami.

5.1.1 Mininet

Mininet je sietovým emulátorom, ktorý dokáže na jednom počítači vytvoriť sieť virtuálnych koncových staníc, prepínačov, kontrolérov a liniek [33]. Koncové stanice majú štandardný Linux sietový softvér a prepínače podporujú OpenFlow pre tvorbu SDN sietí. Mininet podporuje výskum, vývoj, výučbu, prototypovanie, testovanie, ladenie a ďalšie úlohy, ktoré môžu ťažiť z prevádzky celej experimentálnej siete na jednom počítači. Mininet:

- Poskytuje jednoduché a nenáročné sietové testovacie prostredie pre vývoj OpenFlow aplikácií.
- Podporuje regresné testy na systémovej úrovni, ktoré sú opakovateľné a prenositeľné.
- Umožňuje komplexné testovanie topológie, bez potreby zapájania fyzickej siete.
- Obsahuje Command Line Interface (CLI), ktoré si je vedomé topológie a protokolu OpenFlow, vhodné pre ladenie a spúšťanie testov naprieč celou sietou.
- Podporuje ľubovoľné vlastné topológie a obsahuje základnú sadu parametrizovateľných topológií.

- Je ho možné použiť bez programovania, ale zároveň poskytuje priamočiare a rozšíriteľné Python API pre tvorbu sietí a experimentov na nich.

Sieť v emulátore spúšťa skutočný kód zahŕňajúci štandardné Unix/Linux sieťové aplikácie, ako aj skutočné jadro a sieťový set systému Linux. Vďaka tomuto sa dá kód vytvorený a testovaný v tomto prostredí preniesť do skutočnej siete s minimálnymi zmenami. Toto správanie dosahuje používaním odľahčenej virtualizácie, ktorá je založená na sieťových menných priestoroch, pridaných do OS Linux od verzie 2.2.26. Toto umožňuje prezentovanie jedného systému ako celej siete, v ktorej všetky zariadenia pracujú na rovnakom jadre, OS a používateľskom kóde.

5.1.2 Inštalácia emulátora

Emulátor môže byť nainštalovaný buď priamo na hlavnom OS alebo na VM (v tomto prípade bol nainštalovaný na hlavnom OS). Keďže funkčnosť emulátora závisí od jadra systému Linux, použitý bol OS Ubuntu 14.04. Počítač sa nachádzal na IP adrese 192.168.2.50/24. Na oficiálnej stránke emulátora Mininet je poskytnutých niekoľko možností jeho inštalácie. V tomto prípade bola zvolená možnosť jeho natívnej inštalácie zo zdrojových kódov. Tie sa dajú získať pomocou utility *git*, z internetového úložiska programu Mininet príkazom:

```
git://github.com/mininet/mininet
```

Následne sa emulátor a jeho doplnky nainštalujú do aktuálnej zložky príkazom:

```
mininet/util/install.sh -a
```

kde parameter *-a* nainštaluje všetky potrebné súčasti, ako emulátor, virtuálny prepínač Open vSwitch (ktorý je predvoleným prepínačom) a doplnky, ako napríklad modul pre analýzu OpenFlow správ do programu Wireshark. Pre nainštalovanie najnovšej verziu prepínača sa ešte vykonal príkaz:

```
mininet/util/install.sh -V 2.3.0
```

5.1.3 Práca s emulátorom

Celá virtuálna sieť je v emulátore vytvorená následovne. Emulátor spustí jeden root proces, ktorý sa nachádza v mennom priestore OS a v ktorom bude spustený Open vSwitch. V rámci prepínača sa vytvorí požadovaný počet jeho inštancií (podľa požadovaného počtu prepínačov v sieti), do ktorých priradí virtuálne rozhrania, ktoré emulátor vytvoril. Koncové stanice sú vytvorené ako osobitné procesy, pričom pre každý tento proces je vytvorený vlastný

menný priestor a virtuálne rozhranie, cez ktoré bude spojený s vytvoreným prepínačom podľa topológie. Vďaka vlastnému mennému priestoru sa budú vytvorené koncové stanice javiť ako samostatné počítače, pričom ale budú zdieľať spoločné súbory systému. Takto vytvorené počítače sa dajú ovládať prostredníctvom konzoly alebo xterm terminálu. Mininet môže taktiež vytvoriť jednoduchý referenčný OpenFlow kontrolér alebo sa pripojiť na samostatný, nami vytvorený kontrolér. Mená vytvorených zariadení sú v emulátore: *hx* pre koncové stanice, *sx* pre prepínače a *cx* pre kontroléry, kde *x* je číslo zariadenia začínajúce od 1 (0 v prípade kontroléra).

Ak nie je explicitne požadované ináč, pre virtuálne rozhrania sú náhodne vygenerované MAC adresy a koncové stanice dostanú IP adresu postupne od začiatku siete 10.0.0.0/8 (napr. *h1* bude mať IP adresu 10.0.0.1/8). Mininet štandardne vytvára out-of-band kanál medzi OpenFlow prepínačmi a kontrolérom, teda OpenFlow komunikácia prúdi cez osobitné rozhranie, mimo toku bežných dát. Ak sa Mininet aj kontrolér nachádzajú na rovnakom počítači, vytvorené OpenFlow prepínače s kontrolérom komunikujú cez loopback rozhranie. Ak sa nachádzajú na rozdielnych počítačoch, používajú rozhranie pre prístup do siete.

Po nainštalovaní emulátora sa následne dá jednoducho spúštať virtuálna sieť pomocou parametrizovaného príkazu *mn*, kde napríklad:

```
sudo mn --switch=ovsk,protocols=OpenFlow13 --topo=tree,depth=2,fanout=2  
--controller=remote,ip=192.168.2.100
```

vytvorí sieť pozostávajúcu z Open vSwitch prepínačov, využívajúcich OpenFlow 1.3, ktoré sa pripájajú na náš vlastný, vzdialený kontrolér, nachádzajúci sa na IP adrese 192.168.2.100. Pre komunikáciu cez OpenFlow 1.3 počúva kontrolér na porte 6653. Topológia má stromovú štruktúru. Parameter *depth* určuje hĺbku/úroveň stromu a parameter *fanout* zas koľko vetví bude vychádzať z každého prepínača, pričom koncové stanice sú pripojené na prepínače poslednej úrovne. Sieť teda obsahuje 3 prepínače a 4 koncové stanice.

Po vytvorení siete nám Mininet poskytne CLI, prostredníctvom ktorého môžeme so sieťou pracovať. CLI nám umožňuje testovať konektivitu medzi zariadeniami, spúštať na zariadeniach programy a tiež upravovať sieť za chodu. V tomto CLI môžeme namiesto IP adries jednotlivých počítačov taktiež používať ich mená (teda *h1*, *h2* a pod.) Užitočné sú príkazy ako *mn -h* v terminálovom okne alebo *help* v CLI emulátora, ktoré zobrazia užitočné informácie pre zadávanie príkazov. Na otvorenie osobitnej konzoly pre vytvorené koncové stanice môžeme použiť *xterm*, kde napríklad príkazom *xterm h1* otvoríme samostatný terminál pre počítač *h1*. Pre ukončenie programu je určený príkaz *exit* a dobré je použiť následov-

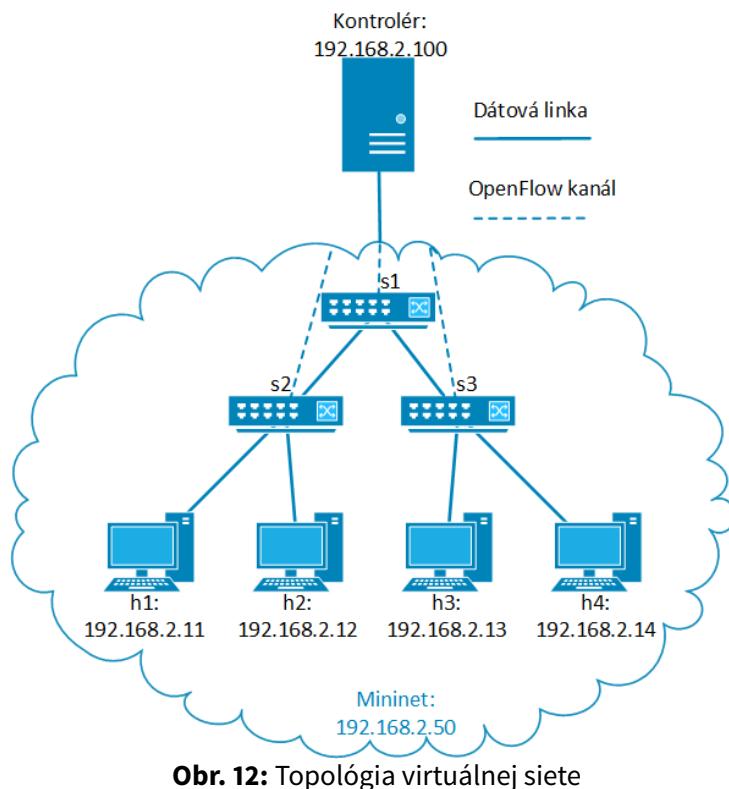
ne príkaz `mn -c`, ktorý vymaže zostatkový stav alebo procesy emulátora, ktoré v systéme ostali.

Mininet je úzko spätý z jazykom Python a ako taký umožňuje cez jeho API vytvárať skripty pre tvorbu vlastných sietí a celkovú automatizáciu všetkých príkazov, ktoré je možné v rámci emulátora používať, ako aj vytváranie nových príkazov.

V topológii, ktorá bola na začiatku tejto sekcie vytvorená by sme mohli pomocou príkazu `py help(h1)` zadanom v CLI emulátora zobraziť všetky metódy, použiteľné pre koncovú stanicu *h1* v rámci Python API. Príkazom `py h1.setIP('192.168.2.11', 24)` by sme napríklad nastavili stanici *h1* danú IP adresu s 24 bitmi dlhou sietovou maskou.

5.1.4 Vytvorenie vlastnej topológie

Pomocou jazyka Python a jeho API pre Mininet sa dá napísati skript, ktorý vytvorí virtuálnu topológiu koncových staníc a virtuálnych Open vSwitch prepínačov používajúcich protokol OpenFlow 1.3, ktoré sa pripoja na nami nastavený SDN kontrolér, nachádzajúci sa na IP adrese 192.168.2.100. Počítače v tejto sieti majú pridelené IP adresy zo siete 192.168.2.0/24. Skript sa nachádza v **prílohe A**. Obrázok nižšie ukazuje tvar vytvorenej siete.



Táto topológiu sa spustí na počítači, kde bol Mininet nainštalovaný a ktorého rozhranie eth0 má IP adresu 192.168.2.50. Prepínače budú posielat OpenFlow správy cez rozhranie

eth0. Každý virtuálny prepínač bude mať osobitný OpenFlow kanál s kontrolérom.

5.2 Fyzická siet

5.2.1 Banana Pi

Cieľom bolo urobiť zo vstavaného počítača Banana Pi hardvérový OpenFlow prepínač. Do počítača sa najprv vložila pamäťová karta s OS Bananian Linux, odľahčená verzia distribúcie Debian 7, ktorá využíva *armhf* repozitáre systému Debian wheezy. Bananian Linux neobsahuje GUI a jeho zámerom je poskytnúť odľahčenú platformu pre malé webové serveri, smerovače, wifi prístupové body a pod. To ho robí ideálnym pre potreby prepínača.

Ako ďalšiu vec bolo potrebné nainštalovať do zariadenia softvérový prepínač Open vSwitch. Inštalácia bola vykonaná prostredníctvom balíčkov. V repozitároch distribúcie sa však v danom čase nachádzali len staré verzie tohto prepínača. Pre použitie najnovšej verzie bolo potrebné vytvoriť inštalačné balíčky ručne. Na to bolo potrebné urobiť nasledovné kroky:

1. `apt-get install build-essential fakeroot`
nainštaluje utility potrebné pre vytvorenie .deb balíčkov.
2. `wget http://openvswitch.org/releases/openvswitch-2.3.0.tar.gz`
stiahne zdrojové kódy prepínača, ktoré treba následne rozbalíť, príkazom
`tar zxvf openvswitch-2.3.0.tar.gz`
a presunúť sa do rozbaleného priečinka príkazom `cd`.
3. Nainštalovať závislosti, ktoré sú uvedené v súbore *debian/control* pod názvom ‘Build-Depends:’ nachádzajúcim sa v danej ceste od aktuálnej zložky. Na ich inštaláciu je vhodné použiť príkaz `apt-get install`. Splnenie tohto kroku si môžeme overiť spustením príkazu `dpkg-checkbuilddeps` v najvyššej úrovni priečinka stiahnutého prepínača. Ten by nemal v prípade splnenia kroku nič vypísať.
4. `fakeroot debian/rules binary`
týmto príkazom spustíme stavbu balíkov prepínača, počas ktorej sa vykoná niekoľko testov. Priebeh potrvá približne 10 minút.
5. Ak príkaz prebehol úspešne, vytvorené .deb súbory sa budú nachádzať v zložke, kde sme umiestnili Open vSwitch zdrojové kódy. Vytvorené balíky sa nainštalujú príkazom:
`dpkg -i openvswitch-common, openvswitch-switch, openvswitch-pki`
(použil som integrovaný Open vSwitch kernel modul zahrnutý v jadre OS).

Pred konfiguráciou je vhodné ubezpečiť sa príkazom `ps -ea | grep ovs`, že procesy programu Open vSwitch sú spustené. Do zariadenia sa následne zapojili kúpené USB-to-Ethernet adaptéry, ktoré bolo treba staticky pridať do konfigurácie sieťových rozhraní. Na zistenie ich názvu v systéme sa dá použiť napríklad príkaz `hwinfo --short`, ktorý zobrazí stručný výpis pripojeného hardvéru. Pomocou príkazov prepínača Open vSwitch následne vytvoríme logický prepínač s názvom *OFSW*, do ktorého pridáme všetky sieťové rozhrania:

```
ovs-vsctl add-br OFSW  
ovs-vsctl add-port OFSW eth0  
ovs-vsctl add-port OFSW eth1  
ovs-vsctl add-port OFSW eth2
```

Do konfiguračného súboru sieťových rozhraní treba staticky pridať *OFSW* ako ďalšie rozhranie a upraviť/pridať konfiguráciu ostatných rozhraní tak, aby fungovali bez IP adres. IP adresu ale bude mať *OFSW*, ktoré bude fungovať aj ako rozhranie pre správu. Konfigurácia všetkých rozhraní v `/etc/network/interfaces` vyzerá nasledovne:

```
auto eth0  
iface eth0 inet manual  
    up ip link set $IFACE up  
    down ip link set $IFACE down  
  
auto eth1  
iface eth1 inet manual  
    up ip link set $IFACE up  
    down ip link set $IFACE down  
  
auto eth2  
iface eth2 inet manual  
    up ip link set $IFACE up  
    down ip link set $IFACE down  
  
auto OFSW  
iface OFSW inet static  
    address 192.168.2.222  
    netmask 255.255.255.0
```

Pri tejto konfigurácii by však nastával problém. *OFSW* je v konfigurácii vedený ako sieťové rozhranie, ktoré sa pri štarte OS Bananian spúšťajú skôr ako samotný Open vSwitch. Pretože by ho v tom momente systém nedokázal rozpoznať, toto rozhranie by nenabehlo a nastavená konfigurácia by nefungovala. Preto treba do súboru `/etc/default/networking` pridať riadok:

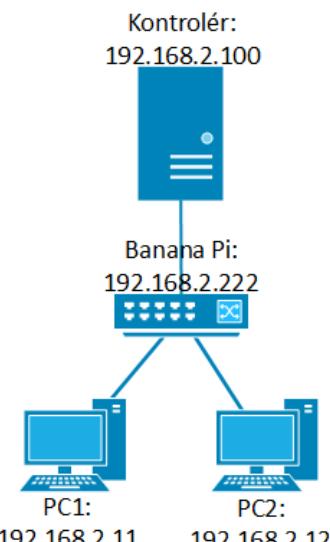
```
EXCLUDE_INTERFACES=OFSW
```

ktorý povie systému aby neskúšal spúštať pri štarte rozhranie *OFSW*. Na jeho spúšťanie využijeme súbor `/etc/rc.local`, ktorý po štarte systému vykoná príkazy v ňom zadané. Pridáme

doňho riadok:

```
ifconfig OFSW up
```

Týmto je Banana Pi pripravené fungovať ako OpenFlow prepínač. Z neho povedie jedným z portov pripojenie na kontrolér a na zvyšné porty sa pripoja dva fyzické počítače ako je to zobrazené na obrázku nižšie.



Obr. 13: Topológia fyzickej siete

Pre použitie verzie 1.3 protokolu OpenFlow je ešte potrebné nastaviť túto verziu ako predvolenú na prepínači príkazom:

```
ovs-vsctl set bridge OFSW protocols=OpenFlow13
```

Neskôr sa na prepínači nastaví pripojenie na kontrolér, ktorý bude v tomto prípade na adrese 192.168.2.100, príkazom:

```
ovs-vsctl set-controller OFSW tcp:192.168.2.100:6653
```

5.3 Zmiešaná siet

Zmiešaná siet v sebe zahŕňa fyzické aj virtuálne zariadenia. Prepínač Banana Pi má opäť jedno z pripojení vedené na kontrolér. Zvyšné dva porty vedú na fyzické počítače. Tentokrát ale jeden z počítačov vďaka programu Open vSwitch zastáva úlohu prepínača. Na tomto počítači sa tiež nachádzajú dva VM, ktoré sú pripojené na Open vSwitch a sú teda súčasťou siete. Tento počítač má v topológii označenie *vSwitch*. Realizovaný bol na Ubuntu 14.04 s GUI, na ktorý sa nainštaloval Open vSwitch vytvorením .deb balíčkov, rovnako ako v prípade zariadenia Banana Pi. V programe Open vSwitch sa vytvorila inštancia virtuálneho prepínača s názvom *OFSW*, do ktorej bolo pridané sieťové rozhranie *eth0* rovnakým spôsobom ako v prípade Banana Pi. Podobná bola aj konfigurácia rozhrania v súbore */etc/network/interfaces*,

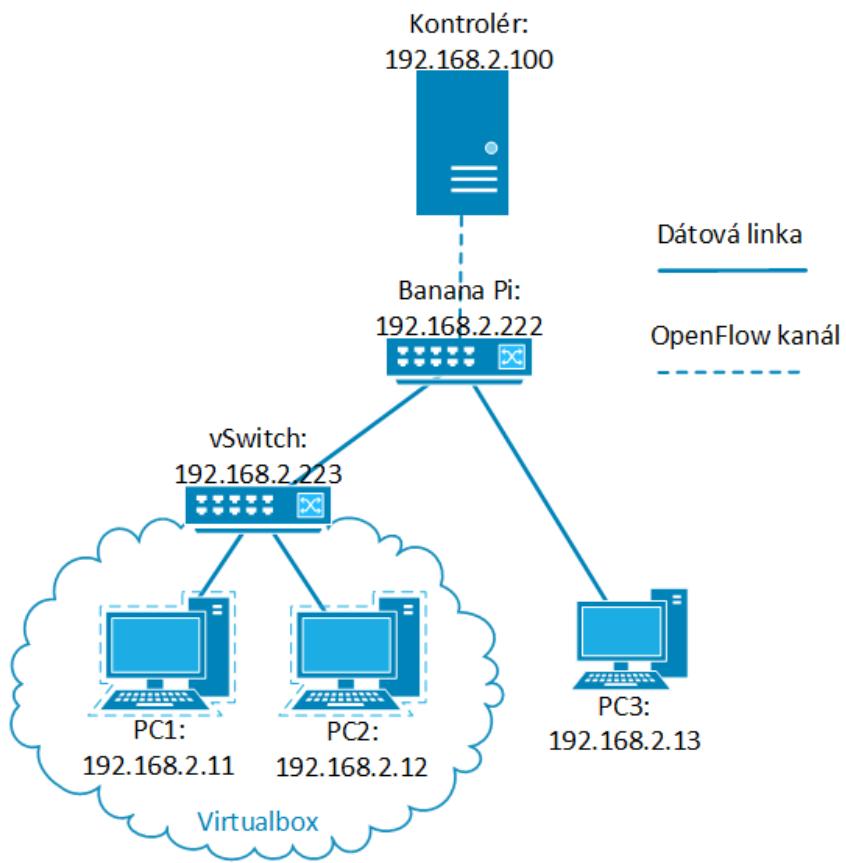
pričom pre vzdialenú správu sa rozhraniu OFSW pridelila IP adresu 192.168.2.223. Pre zavedenie konfigurácie do platnosti bolo vhodné reštartovať rozhrania:

```
sudo ifconfig eth0 down && sudo ifconfig eth0 up  
sudo ifconfig OFSW down && sudo ifconfig OFSW up
```

Následne sa vytvorili dva virtuálne porty *vport1* a *vport2*, ktoré boli pridané do inštancie prepínača. Na tieto porty sa ďalej pripojili virtuálne počítače. Príkazy pre túto časť sú:

```
sudo ip tunctl add mode tap vport1  
sudo ip tunctl add mode tap vport2  
sudo ifconfig vport1 up  
sudo ifconfig vport2 up  
sudo ovs-vsctl add-port OFSW vport1 -- add-port OFSW vport2
```

Ako posledná vec sa do Ubuntu nainštaloval program Virtualbox, v ktorom boli vytvorené dva VM, ktoré zastávali úlohu koncových stanic. V položke sietových rozhraní im bolo potrebné pridať adaptér typu *Bridged Adapter* a v ňom vybrať *vport1* a *vport2*, pre príslušné stanicu. Postup pridávania VM do siete pomocou Open vSwitch je vo firemných riešeniach zautomatizovaný. Virtualizačné systémy ako Xen alebo VMware so zabudovaným Open vSwitch prepínačom, automaticky vytvárajú virtuálne porty a prepájajú vytvorené VM s prepínačom.



Obr. 14: Topológia zmiešanej siete

V zmiešanej sieti je na kontrolér priamo pripojené len Banana Pi. Prepínač vSwitch bude posielat OpenFlow správy kontroléru *in-band*, teda v rámci svojej dátovej linky s Banana Pi.

Rovnako ako u Banana Pi aj tomuto prepínaču bude treba nastaviť použitie OpenFlow 1.3 protokolu príkazom `ovs-vsctl set bridge OFSW protocols=OpenFlow13` a neskôr adresu kontroléra, na ktorý sa ma pripojiť príkazom:

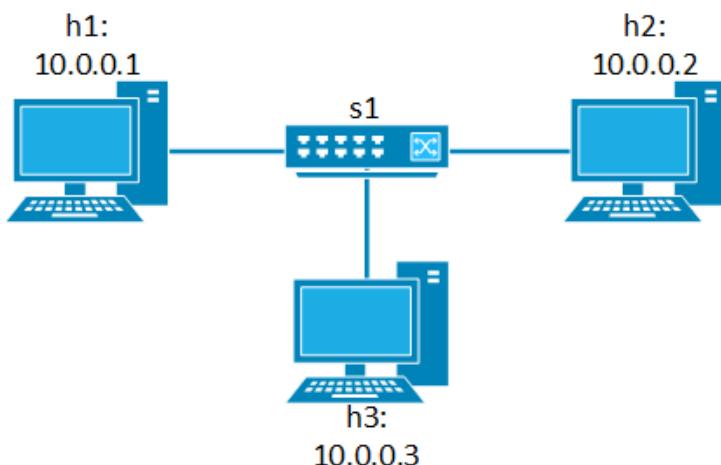
```
ovs-vsctl set-controller OFSW tcp:192.168.2.100:6653
```

5.4 Práca s tokmi v OpenFlow prepínači

Na manipuláciu a kontrolu tokov v OpenFlow prepínačoch je určený SDN kontrolér. Pre ukážku práce s tokmi ich ale môžeme ovládať aj priamo v prepínači, bez zásahu SDN kontroléra. Na túto ukážku si vytvoríme topológiu v emulátore Mininet, kde príkazom:

```
sudo mn --topo=single,3 --controller=none --mac
```

vytvoríme jednoduchú topológiu o jednom OpenFlow prepínači (Open vSwitch) a troch koncových stanicach, pričom prepínač nemá pripojenie na žiadnen kontrolér. Zadaním parametra `--mac` sme zjednodušili MAC adresovanie koncových staníc, kde MAC adresa bude odvodená z ich IP adresy. Teda napríklad pre počítač `h1` s IP 10.0.0.1 bude MAC adresa jeho rozhrania 00:00:00:00:00:01. Vytvorená sieť je zobrazená na obrázku nižšie.



Obr. 15: Jednoduchá topológia

Po vytvorení topológie môžeme v CLI emulátora príkazom `dump` vidieť základné informácie o zariadeniach vytvorenej siete. Následne príkaz `sh ovs-ofctl show s1` vypíše mapovanie názvov portov prepínača `s1` ku ich číslam. Príkazmi, ktoré v CLI emulátora začínajú slovom `sh`, oznamujeme emulátoru aby vykonal príkaz z príkazového riadku OS. Teda zadaním príkazu `ovs-ofctl show s1` priamo na termináli OS by sme dosiahli rovnaký výsledok. Utilita `ovs-ofctl` je Open vSwitch nástrojom pre monitorovanie a správu Open-

Flow prepínačov. Príkaz `ovs-ofctl --help` nám zobrazí možnosti zadávania rôznych parametrov v tomto nástroji. Prepínač momentálne neobsahuje žiadne toky, čo si môžeme overiť zadaním `sh ovs-ofctl dump-flows s1`, ktorý vypíše zoznam všetkých tokov v prepínači. Môžeme ich pridať nasledovným spôsobom:

```
sh ovs-ofctl add-flow s1 action=normal
```

V príkaze nie sú definované podmienky zhody, preto bude tok platiť pre všetky prijaté pakety. Vykonaná akcia je nastavená na *normal*, čo znamená bežné správanie prepínača. Zariadenie sa teraz teda bude správať ako tradičný prepínač, čo si môžeme aj overiť Mininet príkazom `pingall`, ktorý otestuje vzájomné spojenie medzi všetkými počítačmi siete (v tomto prípade *h1* až *h3*). Teraz by sme mali príkazom `sh ovs-ofctl dump-flows s1` dostať podobný výpis ako je uvedený nižšie:

```
NXST_FLOW reply (xid=0x4):  
  cookie=0x0, duration=340.022s, table=0, n_packets=24, n_bytes=1680,  
  idle_age=330, actions=NORMAL
```

Význam týchto položiek je nasledovný:

- cookie - nejasná dátová hodnota, väčšinou generovaná kontrolérom.
- duration - celkový vek vytvoreného toku.
- table - číslo tabuľky, v ktorej sa tok nachádza.
- n_packets - počet paketov, ktoré tok využili.
- n_bytes - počet bajtov dát, ktoré tok využili.
- idle_age - doba neaktívnosti toku (odkedy tokom prešiel posledný paket).
- actions - vykonané akcie na zhodných paketoch.

Príkazom `sh ovs-ofctl del-flows s1` vymažeme všetky toky nachádzajúce sa v *s1*. Po tomto príkaze nám už príkaz `pingall` zlyhá. Teraz vytvoríme dva toky, ktoré umožnia komunikáciu len medzi *h1* a *h2*. Spojenie medzi nimi umožníme príkazmi:

```
sh ovs-ofctl add-flow s1 dl_src=00:00:00:00:00:01,dl_dst=00:00:00:00:00:02,  
  actions=output:2
```

```
sh ovs-ofctl add-flow s1 dl_src=00:00:00:00:00:02,dl_dst=00:00:00:00:00:01,  
  actions=output:1
```

kde *dl_src* a *dl_dst* predstavujú zdrojovú a cielovú MAC adresu zariadení a *output:1* a *output:2* konkrétné výstupné porty prepínača. V týchto tokoch však ešte nie je zahrnuté pravidlo pre Address Resolution Protocol (ARP), ktorý zariadenia potrebujú pre naučenie sa MAC adres

svojich susedov. Preto treba pridať ďalší tok a to:

```
sh ovs-ofctl add-flow s1 dl_type=0x806,nw_proto=1,actions=flood
```

kde *dl_type* predstavuje pole *EtherType*, ktoré v hlavičke Ethernet rámca označuje typ zabeného protokolu (v tomto prípade ARP). Pole *nw_proto* je operačný kód správy, pričom 1 je označenie pre ARP žiadosť. Akciou *flood* udávame prepínaču aby rozosielal paket všetkými portami okrem toho, na ktorý paket prišiel. Tieto 3 položky toku umožnia komunikáciu medzi *h1* a *h2*, ktorú môžeme príkazom *h1 ping h2* overiť. Program už môžeme vypnúť.

Existuje veľké množstvo položiek, ktorými môžeme v OpenFlow prepínači charakterizovať toky. Okrem predvedeného príkladu umožňuje OpenFlow inšpekciu hlavičiek L3 aj L4 vrstvy, ako aj prepisovanie týchto hlavičiek, prácu s VLAN, QoS a mnoho ďalších vecí. Vyššie verzie protokolu OpenFlow umožňujú porovnávať viac polí v hlavičkách paketov a presnejšiu špecifikáciu tokov. V tejto ukážke sme pracovali s verzou 1.0 protokolu OpenFlow. Pre prácu s OpenFlow 1.3 by sme v prepínači použili rovnaké príkazy, ale všetky by začínali reťazcom *ovs-ofctl -O Openflow13*.

Manuálne zadávanie a správa takýchto pravidiel v prepínačoch je samozrejme náročná. Preto je funkcia a prevedenie SDN kontroléra veľmi dôležité.

6 Kontrolér OpenDaylight Helium

OpenDaylight kontrolér je JVM softvér, fungujúci na ľubovoľnej platforme podporujúcej prostredie Java. Je implementáciou SDN konceptu a využíva nasledujúce nástroje [34]:

- Maven - kontrolér používa program Maven pre jednoduchšie a zautomatizované vystavanie programu. Maven používa súbory *pom.xml*, opisujúce závislosti medzi programovými zväzkami a na popisanie zväzkov, ktoré je potrebné načítať a spustiť.
- OSGi - aplikačný rámec na ktorom je kontrolér postavený. Umožňuje dynamické načítanie zväzkov a JAR balíčkov, ako aj prepojenie zväzkov za účelom výmeny informácií.
- Java rozhrania - používajú sa pre počúvanie na udalosti, špecifikácie a tvarovanie vzorov. Je to hlavný spôsob, ktorým konkrétnie zväzky implementujú spätné volanie funkcií pre udalosti a tiež pre indikovanie vedomia o konkrétnom stave.
- REST API rozhrania - tvoria ich Northbound rozhrania, ako napríklad *topology manager, host tracker, flow programmer, static routing* a pod.
- Karaf - malé spúšťacie prostredie založené na OSGi, ktoré poskytuje odľahčený kontajner pre načítavanie rôznych modulov.

6.1 Základné nastavenie

Kontrolér bol nainštalovaný na samostatný počítač s IP adresou 192.168.2.100/24. V počítači bol použitý systém Ubuntu 14.04. Tento kontrolér je napísaný v jazyku Java a pre jeho spustenie treba najskôr nainštalovať javu verzie 1.7.0_45 a vyššiu:

```
sudo add-apt-repository ppa:webupd8team/java  
sudo apt-get update  
sudo apt-get install oracle-java7-installer  
sudo apt-get install oracle-java7-set-default
```

Na oficiálnej stránke projektu OpenDaylight sa dajú stiahnuť predom vystavané komprimované súbory kontroléra. Táto distribúcia nemá z východzích nastavení povolené žiadne funkčné prvky. Tie sa inštalujú a spúšťajú podľa vlastného výberu. Po stiahnutí a rozbalení súborov kontroléra sa vykonaním príkazu `./bin/karaf` v rozbalenej zložke spúšťa kontrolér. Ten pre prácu s ním poskytuje používateľovi vlastné CLI rozhranie. Helium má zoznam komponentov, ktoré sa doňho dajú v CLI prostredí nainštalovať. Ja som si z nich vybral:

- `odl-l2switch-switch-ui` - vnáša logiku L2 prepínania a podporu sledovania zariadení.

- odl-openflowplugin-flow-services-ui - umožňuje nachádzať a kontrolovať OpenFlow prepínače a topológie medzi nimi.
- odl-restconf - povoľuje REST API prístup do MD-SAL obsahujúcej dátové úložisko.
- odl-dlux-core - webové rozhranie kontroléra využívajúce jeho REST/RESTConf API.
- odl-mdsal-apidocs - ukáže zoznam odhalených Northbound API rozhraní.

Prvky sa v CLI kontroléra nainštalujú príkazom:

```
feature:install odl-l2switch-switch-ui odl-openflowplugin-flow-services-ui
               odl-restconf odl-dlux-core odl-mdsal-apidocs
```

Pre overenie sa dá príkazom `feature:list -i` zobrazíť zoznam všetkých nainštalovaných prvkov. O základnú konektivitu v sieti sa starajú komponenty modulu L2Switch, medzi ktoré sa radí:

- Packet Handler - rozkóduje pakety prichádzajúce na kontrolér a príslušne ich spracuje.
- Loop Remover - odstraňuje slučky v sieti.
- ARP Handler - stará sa o rozkódovanie ARP paketov.
- Address Tracker - učí sa adresy sietových zariadení (MAC aj IP).
- Host Tracker - sleduje polohu zariadení v sieti.
- L2Switch Main - inštaluje toky na každom prepínači v závislosti od sietovej premávky.

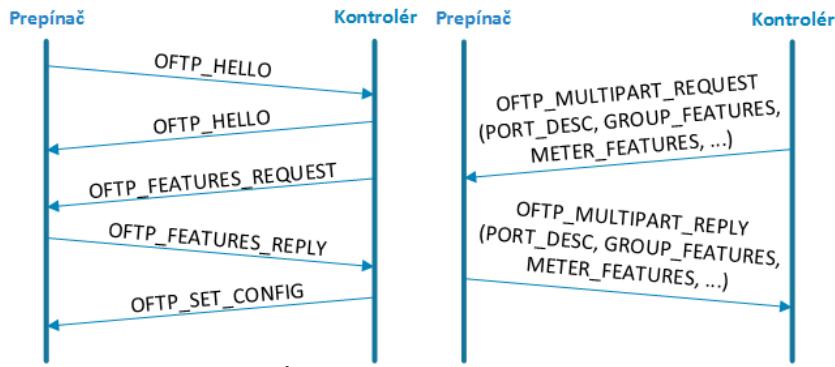
6.2 Funkcionalita protokolu OpenFlow

V tejto časti bude ukázaný spôsob akým komunikuje prepínač a kontrolér pomocou protokolu OpenFlow. Keďže kontrolér je už spustený a nastavený pre základné fungovanie, prijme naň jednoduchú virtuálnu sieť vytvorenú v emulátore Mininet, pozostávajúcu z Open vSwitch prepínača a ku nemu napojených troch koncových staníc. Kontrolér a Mininet boli spustené na osobitných počítačoch, ktoré sa nachádzali v spoločnej sieti 192.168.2.0/24. Sieť bola v emulátore vytvorená príkazom:

```
sudo mn --mac --switch=ovsk,protocols=OpenFlow13
--controller=remote,ip=192.168.2.100 --topo=single,3
```

6.2.1 Počiatočná komunikácia

Po vytvorení kanála medzi kontrolérom a prepínačom nastáva výmena správ, ktorá je ukázaná na obrázku nižšie.



Obr. 16: Úvodná výmena OpenFlow správ

Význam týchto správ je:

- OFTP_HELLO - po vytvorení TCP (alebo TLS) spojenia posiela každá zo strán OFTP_HELLO správu s číslom najvyššej verzie OpenFlow protokolu, ktorú daná strana podporuje.
- OFTP_FEATURES_REQUEST - správu posiela kontrolér prepínaču. Správa nemá telo. Kontrolér si ňou žiada údaje o schopnostiach prepínača.
- OFTP_FEATURES_REPLY - odpoveď prepínača, v ktorej uvádza svoje schopnosti a *Data-path ID*, ktoré identifikuje konkrétné zreťazené spracovanie, ktoré kontrolér v prepínači spravuje. Malo by sa o ňom zmýšľať obdobne ako o MAC adrese tradičného Ethernetového prepínača.
- OFTP_SET_CONFIG - kontrolér v tejto správe posiela sadu bitov, ktorými prepínaču určuje, aké nastavenia má používať a maximálnu veľkosť nového toku, ktorý by mal na kontrolér posielat.
- OFTP_MULTIPART_REQUEST - touto správou si kontrolér žiada aktuálny stav dátového spojenia. Tento typ správy zahŕňa rôzne štatistiky o tokoch, tabuľkách, portoch, frontoch, názve a typu prepínača a pod.
- OFTP_MULTIPART_REPLY - prepínač odpovedá popisom stavu jednotlivých žiadanych prvkov.

6.2.2 Tabuľky tokov prepínača

SDN kontroléry môžu inštalovať OpenFlow toky do prepínačov tromi spôsobmi [35]:

- Reaktívne inštalovanie - keď príde paket na OpenFlow prepínač, ten vykoná vyhľadávanie zhodnej položky v tabuľke tokov. Ak žiadnu zhodu nenájde, odošle paket (alebo jeho časť) správou *packet-in* na kontrolér a očakáva jeho inštrukcie. Reaguje teda na prichádzajúce dáta, konzultuje ich s kontrolérom a vytvára pravidlá v tabuľke tokov na základe jeho inštrukcií. Nedostatok tohto správania je, že spôsobuje oneskorenie

pri prvotnom posielaní paketov, na ktoré prepínač ešte nemá položku v tabuľke tokov ako aj zvýšenú záťaž na prepínač pri väčšom návale takýchto paketov.

- Proaktívne inštalovanie - kontrolér nečaká na *packet-in* správy od prepínača, ale hned na začiatku spojenia inštaluje do prepínača sadu univerzálnych tokov, ktoré pokryjú všetky možné dátá, ktoré na prepínač môžu prísť. Týmto sa postará o to, že *packet-in* správy nikdy nenastanú a eliminuje sa tým oneskorenie spôsobené konzultáciou kontroléra o každom novom toku.
- Hybridné inštalovanie - kombinácia obidvoch predošlých správaní, ktorá umožňuje flexibilitu reakcie na špecifické toky podrobnej kontrolou dátovej prevádzky, pričom si stále zachováva nízke oneskorenie pre ostatné dátá.

Po počiatočnej komunikácii kontroléra s vytvorenou virtuálnou sieťou zobrazíme na prepínači jeho tabuľku tokov. Z nej je možné vidieť, že kontrolér Helium používa hybridný spôsob inštalácie tokov. Výpis tabuľky ukazuje päť tokov, ktoré vyzerajú nasledovne:

```
OFPST_FLOW reply (OF1.3) (xid=0x2):
  cookie=0x2b00000000000000, duration=30.945s, table=0, n_packets=3,
  n_bytes=210, priority=0 actions=drop
  cookie=0x2b00000000000002, duration=27.064s, table=0, n_packets=1,
  n_bytes=70, priority=2, in_port=3 actions=output:2,output:1,CONTROLLER:65535
  cookie=0x2b00000000000001, duration=27.064s, table=0, n_packets=1,
  n_bytes=70, priority=2, in_port=1 actions=output:2,output:3,CONTROLLER:65535
  cookie=0x2b00000000000000, duration=27.064s, table=0, n_packets=1,
  n_bytes=70, priority=2, in_port=2 actions=output:1,output:3,CONTROLLER:65535
  cookie=0x2b00000000000000, duration=30.977s, table=0, n_packets=0,
  n_bytes=0, priority=100, dl_type=0x88cc actions=CONTROLLER:65535
```

Tu je potrebné pripomenúť položku *priority*. Ak sa paket zhoduje z viacerými položkami toku, použije sa tá, ktorá má najväčšiu prioritu. Prvá položka toku slúži ako pravidlo pre zahodenie všetkých paketov, ktoré nenašli zhodu so žiadnou inou položkou toku. Nasledujúce tri položky sú všeobecné pravidlá, pre prenosenie paketov všetkými aktívnymi portami, okrem vstupného. Tieto pakety sa zároveň aj pošlú správou *packet-in* na kontrolér. Kontrolér zároveň správami *packet-out* vysiela z prepínača Link Layer Discovery Protocol (LLDP) pakety. Posledná položka vraví prepínaču aby posielal všetky prichodzie LLDP pakety späť na kontrolér. Týmito správami si kontrolér vybuduje globálny pohľad na sieť. Tieto toky nemajú nastavený časový limit a preto budú v prepínači vždy prítomné.

Teraz otestujeme konektivitu medzi dvoma koncovými stanicami príkazom `h1 ping h2` a následne dáme opäť vypísať tabuľku tokov. Okrem predchádzajúcich piatich tokov v nej pribudli dva nové toky a to:

```

cookie=0x2a00000000000003, duration=1.230s, table=0, n_packets=0,
n_bytes=0, idle_timeout=1800, hard_timeout=3600, priority=10,
dl_src=00:00:00:00:00:01,dl_dst=00:00:00:00:00:02 actions=output:2
cookie=0x2a00000000000002, duration=1.230s, table=0, n_packets=0,
n_bytes=0, idle_timeout=1800, hard_timeout=3600, priority=10,
dl_src=00:00:00:00:00:02,dl_dst=00:00:00:00:00:01 actions=output:1

```

Kontrolér do prepínača nainštaloval dve špecifickejšie položky toku, ktoré preposielajú pakety na základe MAC adres. Toto sa už trochu viac podobá správaniu L2 prepínača. Prítomnosť položiek *idle_timeout* a *hard_timeout* znamená, že tieto pravidlá sú dočasné a po polohodinovej neaktivite tohto toku, respektívne uplynutí jednej hodiny sa tieto položky z tabuľky prepínača odstránia. Teraz môžeme vytvorenú sieť zrušiť.

6.3 Napojenie ukážkových sietí na kontrolér

Kontrolér, ktorý je spustený na počítači s IP adresou 192.168.2.100 je teraz nainštalovaný a nastavený pre poskytovanie hlavných funkcií, čo sme v predošej časti otestovali na jednoduchej topológii, kde bolo vidno ako kontrolér napĺňa tabuľku tokov prepínača. Teraz sa postupne otestuje funkčnosť vytvorených sietí z kapitoly 5.

Ako prvé bola na kontrolér pripojená virtuálna sieť vytvorená cez skript napísaný v jazyku Python, ktorý využíva knižnice pre Mininet. Skript bol pomenovaný *tree_topo.py*. Po presunutí sa do zložky, v ktorej sa tento skript v počítači s emulátorom Mininet nachádza a zadaní príkazu `sudo ./tree_topo.py` sa vytvorila daná topológia. Následne sa zobrazilo CLI emulátora. V ňom bol zadaný príkaz `sh ovs-vsctl show`, ktorý vypíše zoznam všetkých inštancií prepínača (položky Bridge 's1', Bridge 's2' a pod.), portov ku nim pridelených a taktiež stav pripojenia na kontrolér. Pri každej inštancii prepínača by mal byť pri parameetroch spojenia s kontrolérom riadok *is_connected: true*, ktorý značí úspešné pripojenie na kontrolér. Utilita *ovs-vsctl* slúži pre všeobecnú správu Open vSwitch prepínača. Prepínače sa úspešne pripojili na kontrolér a dostali od neho príslušné toky, čím by mala byť dosiahnutá konektivita medzi počítačmi. To sa dalo opäť vyskúšať jednoduchými *ping* alebo *pingall* príkazmi, poprípade otestovaním spojenia medzi počítačmi na šírku pásma pomocou utility *iperf*. Napríklad príkaz `iperf h1 h3` vráti rýchlosť akou sa dajú dátá medzi počítačmi prenášať. Po otestovaní siete sa sieť zrušila a vyskúšaný bol nasledujúci variant siete.

Ako ďalšia bola vyskúšaná fyzická sieť, zahŕňajúca Banana Pi a naň napojené dva počítače. Po zapojení topológie a nastavení IP adres v počítačoch sa Banana Pi pripojilo na kontrolér už spomínaným príkazom `ovs-vsctl set-controller OFSW tcp:192.168.2.100:6653`.

Po zadaní príkazu sa opäť overilo spojenie vo výpise `sh ovs-vsctl show`, ktoré ukazovalo úspešne zostavenie OpenFlow kanála. Vykonaním série testov ako *ping* alebo použitím generátorov sietovej premávky ako D-ITG sa overila správna konektivita medzi počítačmi.

Ako posledná bola overená funkčnosť zmiešanej siete. Po zapojení siete a nastavení IP adres na počítači a dvoch VM sa na obidvoch prepínačoch (Banana Pi a vSwitch) nastavil OpenFlow kanál príkazom:

```
ovs-vsctl set-controller OFSW tcp:192.168.2.100:6653
```

pričom v obidvoch prepínačoch bol následne po príkaze `sh ovs-vsctl show` uvedený stav *is_connected: true*. Vykonané boli testy ako pri fyzickej sieti, ktoré overili konektivitu koncových staníc.

V nasledujúcich častiach práce sa využije vytvorený skript pre virtuálnu sieť na ukázanie správy siete pomocou kontroléra Opendaylight Helium.

6.4 Web rozhranie kontroléra

OpenDaylight User Experience (DLUX) je sietová aplikácia pre správu kontroléra, ktorá využíva protokol OpenFlow. Kontrolér má dva hlavné typy rozhraní: AD-SAL a MD-SAL. Dostáva informácie z rôznych nezávislých modulov cez služby SAL. DLUX taktiež používa SAL pre získanie sietových informácií a ich využitie pri poskytovaní schopností pre sietovú správu.

DLUX môže byť používané ako samostatný program alebo integrovaný v rámci kontroléra. Pre jeho integráciu ho treba nainštalovať ako komponent kontroléra (urobené v časti 6.1). Množstvo prvkov, s ktorými DLUX umožňuje pracovať záleží na tom, aké komponenty sú v kontroléri nainštalované. V tomto príklade sa do rozhrania dá dostať zadáním tohto reťazca do webového prehliadača:

```
http://192.168.2.100:8181/dlux/index.html
```

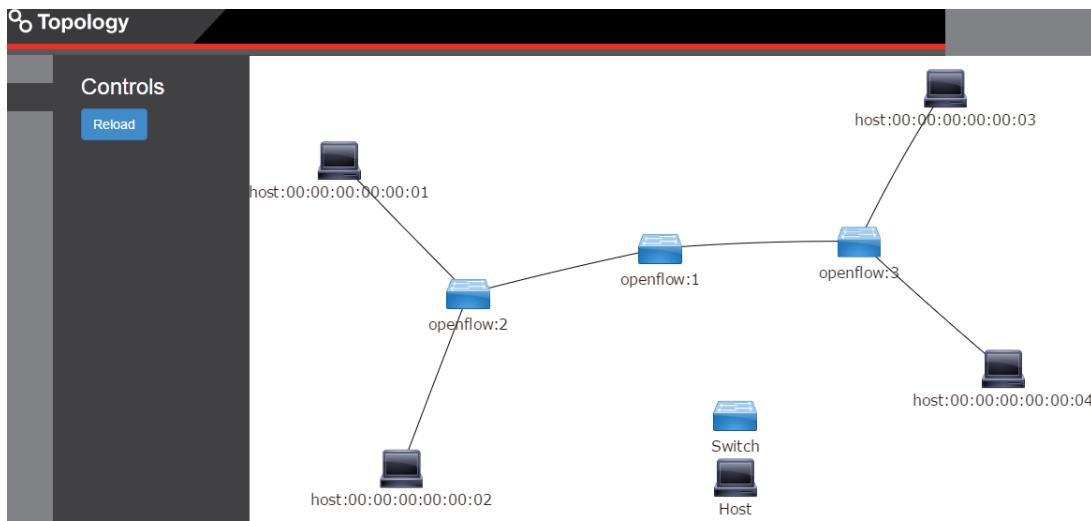
pričom východzie prihlásenie je *admin/admin*. Po prihlásení sa na ľavom paneli zobrazia všetky moduly dostupné pre DLUX. Jednotlivé moduly používajú jedno z rozhraní:

- MD-SAL: *Nodes*, *Topology* a *YANG UI*.
- AD-SAL: *Connection manager*, *Container*, *Network* a *Flows*.

Najpoužívanejšie sú moduly využívajúce MD-SAL. Modul *Nodes* zobrazuje štatistiky siete a informácie o portoch pre jednotlivé prepínače nachádzajúce sa v sieti.

Modul *Network Topology* zobrazuje grafickú reprezentáciu sietovej topológie pomocou protokolu OpenFlow. V topológii sú automaticky zobrazené OpenFlow prepínače, pričom

koncové stanice sú pridané do zobrazenia ak je odhalená ich sietová aktivita.



Obr. 17: DLUX topológia virtuálnej siete z časti 5.1.4

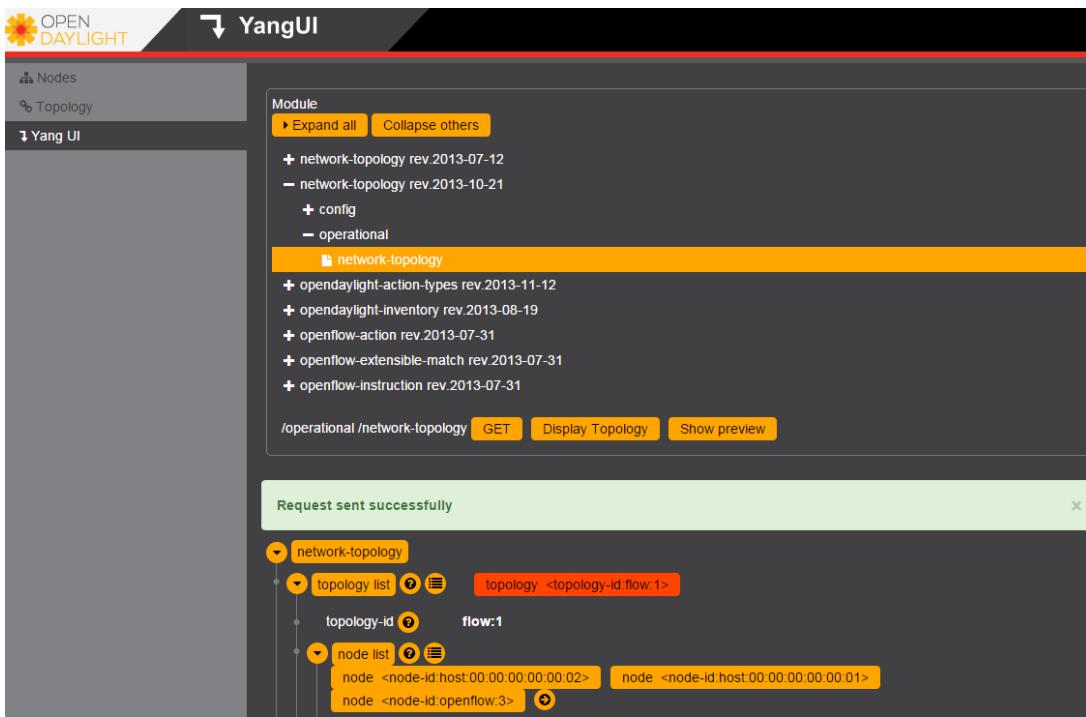
6.4.1 YANG UI

Modul *YANG UI* umožňuje interakciu s kontrolérom. Jeho rozhranie je rozdelené do dvoch častí. V hornej časti zobrazuje strom rozhraní a podrozhraní, ku ktorým je možné pristúpiť a funkcie ktoré je možné pri nich vykonať. Medzi hlavné funkcie patria HTTP metódy *GET*, *PUT*, *POST* a *DELETE*. Ktoré funkcie je možné použiť záleží od podrozhrania, v ktorom sa nachádzame. Je nám umožnený prístup do dvoch dátových úložísk kontroléra a to *config* (obsahujúci dátá vložené cez kontrolér) a *operational* (obsahujúci ostatné dátá).

Spodná časť *YANG UI* zobrazuje možné vstupy, podľa vybraného podrozhrania. Každé takéto podrozhranie predstavuje zoznam prvkov a tie zoznam svojich údajov.

Cez *YANG UI* si napríklad môžeme zobraziť statickú topológiu spustenej siete. Z možných rozhraní si zvolíme *network-topology<revízne číslo topológie>->operational->network-topology* a metódou *GET* požiadame o vrátenie výpisu s topológiou. Tlačidlom *Display Topology* zobrazíme výpis v grafickej podobe, poprípade tlačidlom *Show preview* v podobe čistého textu.

Nainštalovanie modulov využívajúcich rozhranie AD-SAL by rozšírilo ponuku funkcií v DLUX. V dobe písania tejto práce však nastáva pri inštalácii týchto modulov konflikt s už nainštalovanými modulmi, ktorý momentálne znemožňuje využiť tieto moduly.



Obr. 18: DLUX modul YANG UI

6.5 Práca s tokmi cez kontrolér

6.5.1 Postman

Ako bolo v predošlých častiach ukázané, kontrolér OpenDaylight Helium možno ovládať prostredníctvom RESTful rozhraní, ktoré poskytuje. *YANG UI* a *API Explorer* jednoducho využívajú tieto rozhrania pre komunikáciu s kontrolérom. To znamená že pre ovládanie funkcií kontroléra môžeme použiť ľubovoľnú aplikáciu, ktorá rozumie REST API.

Jednou z nich je napríklad REST klient Postman [36], určený pre stavbu, testovanie a dokumentovanie rozhraní. Je odporúčaný OpenDaylight komunitou a umožní nám vytvoriť a poslať ľubovoľné HTTP požiadavky na RESTful rozhranie kontroléra. Je to dobrý spôsob, ako sa naučiť pracovať s kontrolérom prostredníctvom tohto rozhrania. Postman sa inštaluje ako rozšírenie do prehliadača Google Chrome.

6.5.2 Posielanie požiadaviek cez REST API

Súčasný spôsob pridávania tokov do prepínača cez kontrolér pozostáva z týchto krokov:

1. Vytvoríme MD-SAL model toku a pošleme ho na úložisko kontroléra (dataStore).
2. Správca smerovacích pravidiel bude notifikovaný a zavolá príslušnú procedúru (addFlow) pre konkrétnego poskytovateľa služby (ak taký pre daný bod siete existuje).

3. Poskytovateľ (v tomto prípade zásuvný modul) premení tok modelovaný podľa MD-SAL na model toku, ktorý je vhodný pre OpenFlow API.
4. Takýto tok je potom poslaný do OpenFlow knižnice.
5. OpenFlow knižnica zakóduje tok podľa konkrétnej verzie protokolu na dátovej linke a pošle ho na konkrétny prepínač, ktorý si ho po prijatí vloží do tabuľky.

6.5.3 Blokovanie IPv4 komunikácie

Na spustený kontrolér teraz pripojíme virtuálnu sieť z vytvoreného skriptu (5.1.4) a ukážeme prácu s tokmi v tejto sieti posielaním HTTP požiadaviek pomocou REST klienta Postman.

Povedzme že chceme zakázať IPv4 komunikáciu medzi počítačmi *h1* a *h4* na základe IP adresy a chceme tak urobiť na prepínači *s1*. Na toto je potrebné pridať jeden tok pre každý smer. Začneme s tokom pre zákaz IPv4 správ z *h1* na *h4*.

Na hlavnom panely klienta Postman si vytvoríme novú správu následovne:

1. Do URL poľa zadáme reťazec `http://192.168.2.100:8181/restconf/config/opendaylight-inventory:nodes/node/openflow:1/table/0/flow/10` ktorý značí že požiadavka sa týka prepínača, ktorého ID v kontroléri je ‘openflow:1’ a konkrétnie jeho tabuľky s číslom 0 a toku s číslom 10.
2. Z ponúknutých HTTP metód vyberieme metódu *PUT*, určenú pre vkladanie tokov.
3. Teraz vyplníme potrebné hlavičky správy, ktoré sú:
 - Content-Type = application/xml
 - Accept = application/xml
 - Authentication = Basic
 - Authorization = Basic YWRtaW46YWRtaW4=

V poslednej hlavičke odosielame na kontrolér svoje prihlásovacie údaje (*admin/admin*).

Nie je povinná, ale bez nej by sme boli o ne vždy pri odoslaní správy požiadani.

4. Ostáva ešte vyplniť telo správy, kde zvolíme možnosť *raw* a vyberieme formát *XML*.

Do tela správy vložíme tento model toku zapísaný v XML:

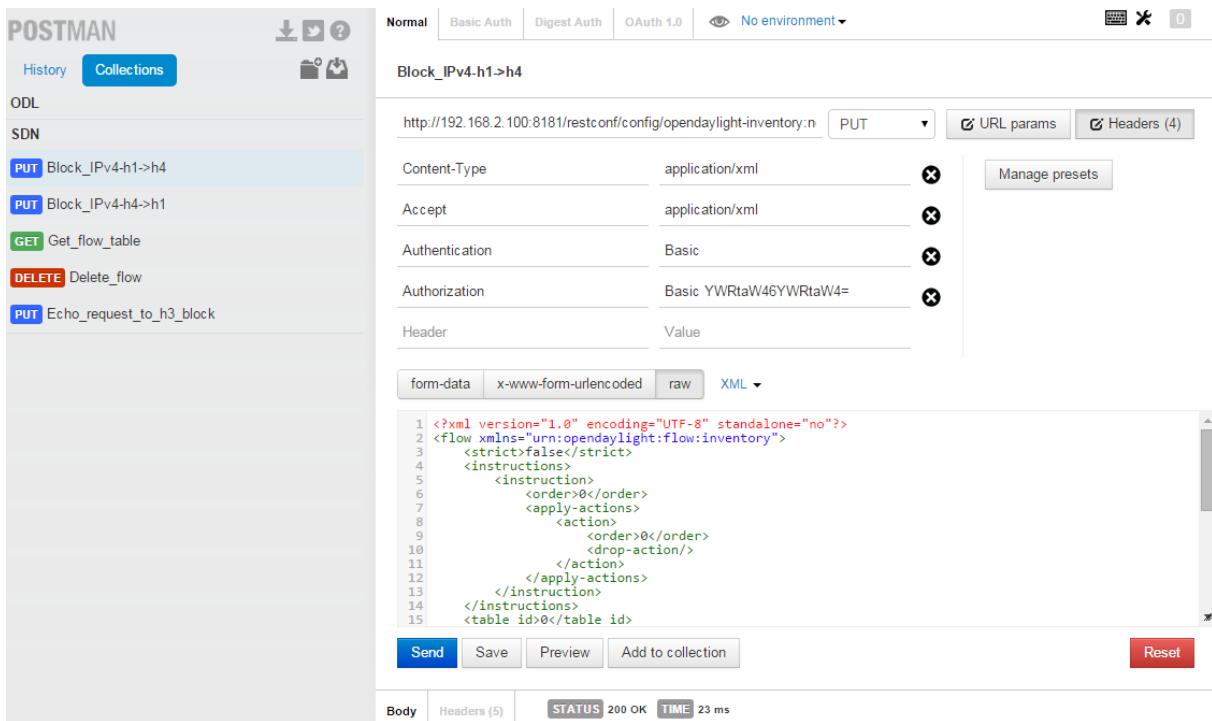
```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<flow xmlns="urn:opendaylight:flow:inventory">
  <strict>false</strict>
  <instructions>
    <instruction>
      <order>0</order>
```

```

        <apply-actions>
            <action>
                <order>0</order>
                <drop-action/>
            </action>
        </apply-actions>
    </instruction>
</instructions>
<table_id>0</table_id>
<id>10</id>
<cookie_mask>255</cookie_mask>
<match>
    <ethernet-match>
        <ethernet-type>
            <type>2048</type>
        </ethernet-type>
    </ethernet-match>
    <ipv4-source>192.168.2.11/32</ipv4-source>
    <ipv4-destination>192.168.2.14/32</ipv4-destination>
</match>
<hard-timeout>3600</hard-timeout>
<cookie>7</cookie>
<idle-timeout>1800</idle-timeout>
<flow-name>Block_IPv4-h1->h4</flow-name>
<priority>20</priority>
<barrier>false</barrier>
</flow>

```

Na začiatku, v časti *instructions* udávame akciu aká sa má na paketoch zhodných s tým-to tokom vykonať. V tomto prípade to je akcia *drop-action*, teda pakety sa zahodia. Položka *table_id* a *id*, označujú číslo tabuľky a číslo toku v nej, pričom sa musia zhodovať s tými uvedenými v URL poli. V časti *match* udávame všetky polia, v ktorých sa pakety musia zhodovať aby nastala zhoda. Pole *ethernet-type* s hodnotou 2048 označuje IPv4. Ak vykonávame porovnanie hlavičiek paketov aj na protokoloch vyšších vrstiev (ako napríklad IPv4), vždy je potrebné v predchádzajúcich protokoloch uviesť aký typ protokolu sa v ňom nachádza (ako *EtherType* hodnota v Ethernet rámcu označujúca IPv4 ako ďalší protokol). V opačnom prípade by prepínač ignoroval pravidlá zhody nastavené pre protokoly vyšších vrstiev. Pod poľom *ethernet-type* sú uvedené polia pre zdrojovú a cieľovú IPv4 adresu paketa. Pre účinnosť toku musí mať pole *priority* najvyššiu hodnotu, z tokov v prepínači, pre ktoré by tiež nastala zhoda s paketmi, ktoré chceme zablokovať.



Obr. 19: Tvorenie HTTP správ v prostredí Postman

Po poslaní takejto správy by nám mala z kontroléra prísť odpoveď 200 OK. Tento tok budeme teraz vidieť aj v tabuľke tokov na prepínači *s1*. Príkaz ping medzi *h1* a *h4* už nebude fungovať. IPv4 pakety z *h4* na *h1* však stále môžu prejsť, čo sa dá overiť zachytením paketov. V CLI emulátora si otvoríme terminály pre dané počítače príkazmi `xterm h1` a `xterm h4`. Na termináli *h1* zadáme príkaz `tcpdump -nni h1-eth0 icmp`, ktorý spustí počúvanie pre Internet Control Message Protocol (ICMP). Keď spustíme z terminálu *h4* `ping 192.168.2.11` uvidíme, že *h1* dostáva a odpovedá na prichodzí ping. Počítač *h4* odpovede už nedostane kvôli pridanému toku.

Pridajme teraz aj pravidlo pre druhý smer. Na toto stačí pozmeniť vytvorenú správu, kde v URL poli zmeníme na konci číslo toku z 10 na 11 a v tele správy urobíme takéto úpravy:

- pole *id* nastavíme na hodnotu 11 (ID toku).
- zameníme hodnoty v *ipv4-source* a *ipv4-destination*.
- *cookie* navýšime na hodnotu 8.
- Do *flow-name* vložíme ‘*Block_IPv4-h4->h1*’

Odoslaním takejto správy kontrolér nainštaluje do prepínača ďalší tok, ktorým docielime obojstranné blokovanie IPv4 komunikácie medzi *h1* a *h4*. Takto vyzerajú pridané toky na *s1*:

```
cookie=0x7, duration=49.426s, table=0, n_packets=1, n_bytes=98,
idle_timeout=1800, hard_timeout=3600, priority=20, ip,nw_src=192.168.2.11,
```

```
  nw_dst=192.168.2.14 actions=drop  
    cookie=0x8, duration=46.524s, table=0, n_packets=1, n_bytes=98,  
    idle_timeout=1800, hard_timeout=3600, priority=20, ip,nw_src=192.168.2.14,  
    nw_dst=192.168.2.11 actions=drop
```

6.5.4 Výpis tokov cez kontrolér

Výpis z tabuľky tokov prepínača si môžeme vyžiadať aj od kontroléra, zaslaním HTTP požiadavky *GET*. Pre výpis z tabuľky číslo 0 z prepínača *s1* (v kontroléri vedeného ako *openflow:1*) treba zadať do URL poľa správy `http://192.168.2.100:8181/restconf/operational/opendaylight-inventory:nodes/node/openflow:1/table/0/` Metódu nastavíme na *GET* a pridáme hlavičku *Authorization* s hodnotou, ktorá bola uvedená v predchádzajúcom príklade. Môžeme zadať aj nepovinnú hlavičku *Accept* s hodnotou '*application/xml*' ak chceme vrátiť výpis vo forme XML (ináč vracia výpis formátu JSON). Telo bude prázdne. V tomto výpise sa budú nachádzať aj naše dva toky, ktoré sme na prepínač pridali.

6.5.5 Odstránenie tokov cez kontrolér

Kontroléru môžeme takisto zaslať požiadavku na vymazanie tokov, prípadne celých tabuľiek tokov z prepínačov. Ku tomuto sa dá využiť HTTP metóda *DELETE*. Pre vymazanie tokov, ktoré sme vytvorili treba skonštruovať správu typu *DELETE*, opäť s hlavičkou *Authorization*, pričom jej telo bude prázdne. Pre vymazanie prvého z našich tokov treba zadať do správy URL: `http://192.168.2.100:8181/restconf/config/opendaylight-inventory:nodes/node/openflow:1/table/0/flow/10`

Na zmazanie druhej správy stačí zmeniť číslo toku na konci URL na hodnotu 11. Bez udania URL s časťou `flow/<id>` by sme zmazali všetky pridané toky z danej tabuľky (ostali by len toky automaticky generované kontrolérom).

6.5.6 Blokovanie ICMP správ

Tentokrát povedzme že nechceme aby počítač *h3* bolo možné v tejto sieti objaviť cez utilitu ping. Zároveň však chceme aby *h3* mohlo používať ping na zistenie prítomnosti ostatných počítačov. V tomto prípade nasadíme pravidlo na prepínač *s3*, ktoré bude zahadzovať prichádzajúce ICMP správy typu *Echo Request* na počítač *h3*.

V klientovi Postman si opäť vytvoríme novú správu typu *PUT*, tentokrát z URL adresou `http://192.168.2.100:8181/restconf/config/opendaylight-inventory:nodes/`

node/openflow:3/table/0/flow/15

a pridáme štyri hlavičky:

- Content-Type = application/xml
- Accept = application/xml
- Authentication = Basic
- Authorization = Basic YWRtaW46YWRtaW4=

XML telo správy bude vyzerat takto:

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<flow xmlns="urn:opendaylight:flow:inventory">
    <strict>false</strict>
    <table_id>0</table_id>
    <id>15</id>
    <cookie_mask>255</cookie_mask>
    <match>
        <ethernet-match>
            <ethernet-type>
                <type>2048</type>
            </ethernet-type>
        </ethernet-match>
        <ipv4-source>192.168.2.0/24</ipv4-source>
        <ipv4-destination>192.168.2.13/32</ipv4-destination>
        <ip-match>
            <ip-protocol>1</ip-protocol>
        </ip-match>
        <icmpv4-match>
            <icmpv4-type>8</icmpv4-type>
            <icmpv4-code>0</icmpv4-code>
        </icmpv4-match>
    </match>
    <hard-timeout>3600</hard-timeout>
    <cookie>9</cookie>
    <idle-timeout>1800</idle-timeout>
    <flow-name>Block_echo_requests_to_h3</flow-name>
    <priority>30</priority>
</flow>
```

V tomto toku chýba časť *instructions*, ktorá udávala akú akciu má prepínač so zhodnými paketmi vykonať. Špecifikácia protokolu OpenFlow udáva, že pri absencii tejto časti má prepínač zhodný paket zahodiť, keďže nemá akciu, ktorú by s ním vykonal. Povinné pole *ethernet-type* je opäť nastavené na 2048, označujúc IPv4 ako ďalší protokol. Pole *ipv4-source* nájde zhodu s ľubovoľnou IP adresou zo siete 192.168.2.0/24, pričom *ipv4-destination* označuje IP adresu počítača h3. Povinné pole *ip-protocol* s hodnotou 1 označuje, že v IPv4 pakete

sa má nachádzať ICMP správa. Polia *icmpv4-code* s hodnotou 0 a *icmpv4-type* s hodnotou 8 označujú ICMP *Echo Request* správu. Pole *priority* bolo nastavené na hodnotu 30, zabezpečujúcu uplatnenie pravidla pri zhode s paketmi.

Po poslaní správy si môžeme skontrolovať pridanie toku buď cez požiadanie kontroléra o výpis tabuľky tokov z prepínača *s3*, alebo priamo kontrolou tabuľky tokov cez *ovs-ofctl* nástroj v emulátore Mininet. Príkazom *pingall* z CLI emulátora Mininet sa môžeme jednoducho presvedčiť o úspešnom nasadení pravidla:

```
*** Ping: testing ping reachability
h1 -> h2 X h4
h2 -> h1 X h4
h3 -> h1 h2 h4
h4 -> h1 h2 X
*** Results: 25% dropped (9/12 received)
```

Vidíme že žiaden počítač v sieti (okrem samotného *h3*) nedokázal utilitou ping overiť dosiahnutelnosť *h3*. Počítač *h3* však sám dokázal dosiahnuť všetky počítače siete.

Tvorcovia kontroléra OpenDaylight Helium očakávajú, že veľa aplikácií bude využívať toto REST API pre komunikáciu a správu kontroléra. Napísaný program by sám generoval podľa svojej logiky a požiadaviek používateľa správy, ktorými by riadil kontrolér a tým pádom celú SDN siet. Cez toto rozhranie by mohli byť napísané programy, ktoré by fungovali ako dynamický firewall, monitorovacie programy, ktoré by zbierali štatistické údaje z kontroléra, programy upravujúce smerovanie v sieti a rôzne ďalšie.

6.6 Clustering kontroléra

Pojmom clustering označujem mechanizmus umožňujúci viacerým procesom a programom spolupracovať ako jedna entita. Týmto mechanizmom môžeme spojiť viaceré inštancie kontroléra, ktoré budú spolupracovať. Výhody, ktoré takéto zoskupenie prináša sú:

- Škálovateľnosť - viacero kontrolérov dokáže potenciálne vykonať viac práce a dokáže me takto uchovávať viac dát. Dáta môžeme rozložiť na menšie kúsky (črepy) a distribuovať ich naprieč zoskupením alebo vykonávať konkrétné operácie na konkrétnych členoch zoskupenia.
- Vysoká dostupnosť - aj po páde jedného z kontrolérov by ostatné inštancie nadalej pracovali a boli dostupné.
- Perzistentné dáta - nestratia sa žiadne dáta získané kontrolérom ani po jeho reštarte alebo páde.

Clustering môže byť vykonaný v rámci jedného OpenDaylight kontroléra alebo viacerých kontrolérov.

6.6.1 Clustering na jednom kontroléri

Pre využitie tohto mechanizmu je potrebné pri inštalácii komponentov kontroléra nainštalovať ako prvý *odl-mdsal-clustering*. Tiež je potrebné nainštalovať zväzok *Jolokia* príkazom:

```
install -s mvn:org.jolokia/jolokia-osgi/1.1.5
```

Toto nám pridá vlastnosti ako:

- Delenie dát - MD-SAL strom, uchovávaný v pamäti sa rozloží na menší počet podstrovov (*inventory, topology a default*)
- Perzistentné dáta - všetky dáta jednotlivých črepín sú uložené na disku, čo umožní ich obnovenie aj po reštarte alebo páde kontroléra.

6.6.2 Clustering na viacerých kontroléroch

Pri zoskupení viacerých kontrolérov je potrebné použiť aspoň tri kontroléri pre zaistenie vysokej dostupnosti. Keby sme nastavili clustering len s dvoma členmi, pri páde jedného z nich by už kontrolér nebol schopný ďalej fungovať.

Začiatočná konfigurácia je rovnaká ako pri využití mechanizmu na jednom kontroléri, pričom tento postup sa zopakuje u každého z členov. Teraz treba na každom členovi upraviť dva súbory, nachádzajúce sa v zložke kontroléra. Prvým je *configuration/initial/akka.conf* kde:

- v každom riadku kde je reťazec *hostname* treba zmeniť IP adresu 127.0.0.1 za IP adresu konkrétneho zariadenia.
- v riadku s položkou *seed-nodes* treba uvedeným zápisom vymenovať v rámci zátvoriek všetkých členov zoskupenia, pričom IP adresu 127.0.0.1 zameniť za IP adresy členov.
- v sekcií *roles* treba každému členovi určiť unikátne meno. Napríklad prvý člen bude *member-1*, druhý *member-2* a pod.

Následne v súbore *configuration/initial/module-shards.conf* treba v sekcií *replicas* udať unikátne meno člena, ktorého aktuálne nastavujeme. Pre využitie vysokej dostupnosti treba v každej tejto sekcií udať mená všetkých členov zoskupenia.

Na záver treba ešte na každom členovi zoskupenia zadať tento príkaz:

```
JAVA_MAX_MEM=4G JAVA_MAX_PERM_MEM=512m ./karaf
```

Teraz dokáže byť kontrolér OpenDaylight spustený v zoskupení viacerých členov. Na prístup ku dátovému úložisku sa dá použiť ľubovoľný z členov zoskupenia.

6.7 Rozšírenie funkcionality kontroléra

6.7.1 Technický prehľad architektúry kontroléra

Ako už bolo spomenuté, OpenDaylight je modulárной platformou, v ktorej väčšina modulov využíva sadu už existujúcich základných služieb a rozhraní. OpenDaylight ako aj väčšina ostatných kontrolérov poskytuje určité prirodzené funkcie pre zabezpečenie kľúčových schopností, ako:

- Schopnosť počúvať pre asynchronné udalosti (ako *PACKET_IN*, *FLOW_REMOVED* a pod.) a registrovanie spätných funkcií (ako *receiveDataPacket* a pod.)
- Schopnosť analyzovať príchodzie pakety (ARP, ICMP a pod.) a vytvárať pakety na poslanie do siete.
- Schopnosť vytvárať/posielat SDN/OpenFlow správy programovateľnej dátovej úrovni.
- Schopnosť spravovať rozhodovacie moduly v rámci kontroléra.

V kontroléri Opendaylight sa tieto schopnosti dosiahnu kombináciou tried implementujúcich SAL a rozhraní pre počúvanie.

Kontrolér poskytuje niekoľko základných zväzkov, z ktorých každý exportuje dôležité služby pomocou Java rozhraní [37]. Medzi niektoré z nich, ktoré sú užitočné pri vývoji sieťových služieb patria zväzky z nasledujúcej tabuľky.

zväzok	poskytnuté rozhranie	popis
arphandler	IHostFinder	Komponent zodpovedný za učenie sa polohy koncových staníc zaobchádzaním s ARP.
hosttracker	IFlptoHost	Sleduje polohu koncových staníc vzhľadom na SDN sieti.
switchmanager	ISwitchManager	Komponent obsahujúci inventár s informáciami o všetkých známych prepínačoch v sieti.
statisticsmanager	IStatisticsManager	Komponent používajúci SAL ReadService pre zber viacerých štatistik o SDN sieti.
topologymanager	ITopologyManager	Komponent obsahujúci graf celej siete.
sal	IReadService	Rozhranie vracajúce hardvérový pohľad na tok/port/front prepínača siete.
sal	ITopologyService	Obsahuje topologické metódy, ktoré SAL poskytuje aplikáciám.
sal	IFlowProgrammerService	Rozhranie pre inštaláciu/úpravu/mazanie tokov na prepínači v SDN sieti.
sal	IDataPacketService	Služba obsluhujúca dátové pakety, ktorú SAL poskytuje aplikáciám.

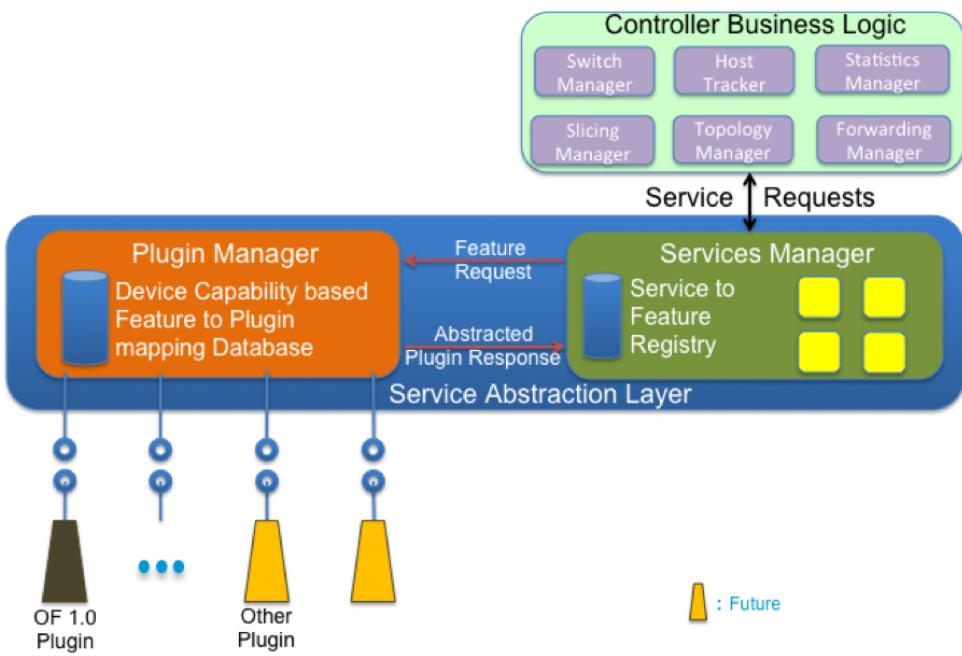
Tabuľka 4: Zoznam užitočných OpenDaylight Java rozhranií

Moduly môžu súperiť o rovnakú udalosť, ktorú by chceli obslužiť. Toto vyžaduje koordináciu naprieč modulmi pre zabránenie konfliktom. Modul môže kontroléru signalizovať, keď ukončí spracovanie paketu. Po spracovaní paketu modul vráti jeden z týchto stavov na SAL:

- *PacketResult.CONSUME* - paket bol spracovaný a žiaden iný modul by ho nemal mať.
- *PacketResult.KEEP_PROCESSING* - paket bol spracovaný, ale stále môže byť poslaný ďalším modulom.
- *PacketResult.IGNORED* - Paket bol ignorovaný a môže byť poslaný na ostatné moduly.

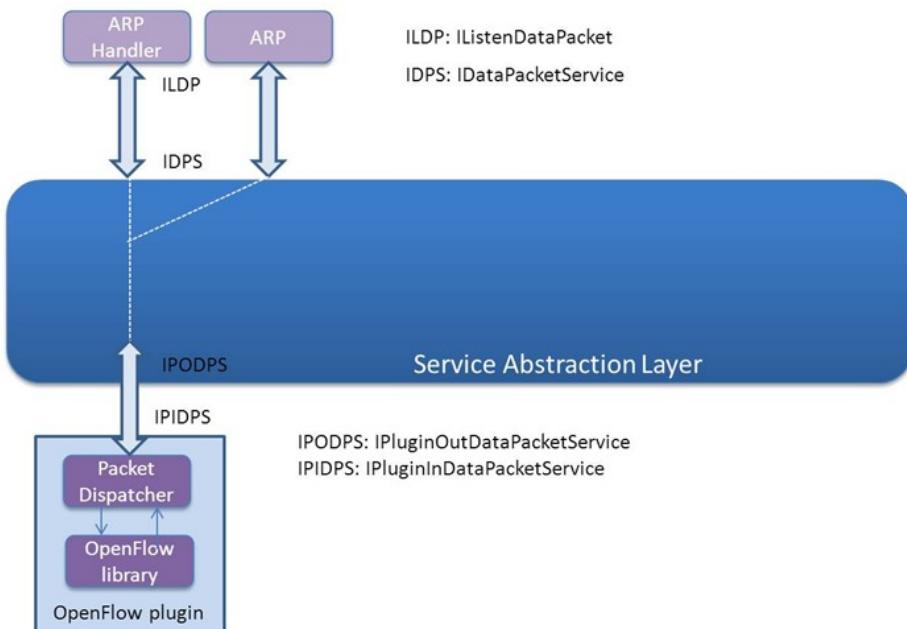
6.7.2 Service Abstraction Layer

SAL je srdcom modulárneho dizajnu kontroléra, pretože umožňuje podporu viacerých South-bound protokolov a poskytuje konzistentné služby pre moduly a aplikácie [38].



Obr. 20: Service Abstraction Layer [38]

OSGi aplikačný rámec umožňuje dynamicky prepojiť zásuvné moduly pre stále vyvíjajúce sa Southbound protokoly. SAL poskytuje základné služby ako *Device Discovery*, ktoré sú používané modulmi ako *Topology Manager* pre vystavanie topológie a schopností zariadení. Služby sú vystavané použitím funkcií poskytnutých zásuvnými modulmi (v závislosti na prítomnosti modulu a schopnosti sietového zariadenia). Na základe požiadavky služby vrstva SAL mapuje vhodný zásuvný modul a tým pádom použije najvhodnejší Southbound protokol pre interakciu s daným sietovým zariadením. Na obrázku nižšie je príklad fungovania SAL.



Obr. 21: Data Packet Service [38]

Popis jednotlivých funkcií z obrázku vyššie je nasledujúci:

- `IListenDataPacket` - služba implementovaná modulom alebo aplikáciou na vyšej vrstve, ktorá chce prijímať dátové pakety (ARP Handler je jedným z takých modulov).
- `IDataPacketService` - toto rozhranie je implementované vrstvou SAL a poskytuje službu posielania a prijímania paketov z agenta. Táto služba bude registrovaná v OSGi registri služieb aby ku nej aplikácia mohla získať prístup.
- `IPluginOutDataPacketService` - toto rozhranie je exportované vrstvou SAL, keď chce zásuvný modul protokolu doručiť paket na aplikačnú vrstvu.
- `IPluginInDataPacketService` - toto rozhranie je exportované zásuvným modulom protokolu a je použité pri odosielaní paketu cez SAL na agenta, nachádzajúceho sa na sietovom zariadení.

Teraz sa pozrime ako by bol vykonaný kód v SAL spolu so službami a zásuvnými modulmi:

1. Povedzme že OpenFlow zásuvný modul dostane ARP paket, ktorý je potrebné doručiť aplikácii *ARP Handler*.
2. OpenFlow zásuvný modul zavolá `IPluginOutDataPacketService` pre poslanie paketu na vrstvu SAL.
3. Aplikácia *ARP Handler* by bola registrovaná v službe `IListenDataPacket`. Po prijatí paku- tu (z bodu 2.) by vrstva SAL odovzdala paket aplikácií *ARP Handler*.
4. Aplikácia teraz môže paket spracovať.

Pre opačný smer, kedy by aplikácia posielala paket von, by vykonanie prebehlo takto:

1. Aplikácia vytvorí paket a zavolá rozhranie `IDataPacketService`, poskytované vrstvou SAL pre odosielanie paketov. Zariadenie cieľovej siete musí byť poskytnuté ako súčasť API.
2. SAL následne zavolá rozhranie `IPluginInDataPacketService` pre daný zásuvný modul protokolu na základe zariadenia cieľovej siete (v tomto prípade pre OpenFlow modul).
3. Zásuvný modul protokolu potom odošle paket na príslušné sietové zariadenie. Modul sa postará o všetky procedúry špecifické pre daný protokol.

6.7.3 Možnosti programovania kontroléra

Pre programovanie kontroléra je potrebné mať nainštalované:

- Java JDK verzie 1.7.
- Maven verzie 3.1.1 a vyššie - nástroj pre automatickú stavbu programu, hlavne určený pre projekty jazyka Java.

- Vývojové prostredie Eclipse pre Java EE vývojárov so zásuvným modulom Xtend a M2Eclipse 1.3 (Maven integrácia pre Eclipse).

Následne sa dajú stiahnuť zdrojové kódy kontroléra, zvyčajne použitím príkazu:

```
git clone https://git.opendaylight.org/gerrit/controller
```

ktorý stiahne kódy z oficiálnych repozitárov projektu Opendaylight. Tie sa po stiahnutí musia skompilovať nástrojom *Maven* a výsledné súbory je možné importovať ako projekt do prostredia Eclipse, kde je možné kód upravovať a napísať nové moduly, ktoré chceme v kontroléri implementovať.

Dôležité je spomenúť, že existujú dva typy SAL:

- API-driven SAL (AD-SAL), kde je API staticky definované pri komplikácii/zostavení, pričom ‘poskytovateľ’ a ‘spotrebiteľ’ udalostí, respektíve dát, sú priamo prepojený medzi sebou v kóde.
- Model-driven SAL (MD-SAL), kde je API generované z modelov počas komplikácie a načítané do kontroléra, keď sa do kontroléra načíta zásuvný modul OSGi zväzku. V MD-SAL prechádzajú všetky udalosti a dáta od ‘poskytovateľa’ ku ‘spotrebiteľovi’ cez centrálné dátové úložisko.

AD-SAL je pôvodný spôsob akým sa spájali Northbound a Southbound rozhrania. V ňom sa napísané moduly musia spúštať, respektíve ukončovať pomocou takzvaných *Bundle-Activator* tried. Keďže všetok kód sa píše manuálne, rastie šanca pre možné chyby v kóde a úpravy modulov môžu byť relatívne náročné.

MD-SAL je nový prístup, ktorý používa YANG modelovací jazyk ako jazyk pre modelovanie konfigurácie, závislostí a ktorý udáva dáta pre moduly. Z takto vytvorených modelov vygenerujú YANG nástroje Java rozhrania, ktoré sa zviažu nástrojom *Maven* do OSGi zväzkov. MD-SAL takisto poskytne RESTful rozhrania pre vytvorené zväzky.

Tvorba aplikácií sa tým pádom líši v závislosti od toho, ktorý typ SAL sa pri tvorbe využije. Každý z typov sa líši vo vlastnostiach, ako sú ľahkosť používania, škálovateľnosť, jednoduchosť hľadania chýb a sada dostupných funkcií. Pri tvorbe aplikácie sa môžu použiť obidva typy SAL.

6.7.4 API Explorer

MD-SAL RESTful rozhrania sú navrhnuté podľa protokolu RESTCONF. Sú dynamicky generované za behu programu na základe YANG modelov definujúcich ich dátá. Keďže sú rozhra-

nia dynamicky generované, neexistujú tu staticky kompliované triedy jazyka Java, ktoré by spoločne s komentárm poskytli dobrú dokumentáciu. Tento nedostatok dokumentácie bol donedávna nevýhodou využívania MD-SAL rozhraní.

Z tohto dôvodu bol vytvorený RESTCONF API Explorer [39], ktorý rieši tento nedostatok. Nasadzuje OSGi zväzok, ktorý generuje a poskytuje dokumentáciu za behu programu. V po- užitej verzii kontroléra je dostupný prostredníctvom webového prehliadača na adrese:

<http://192.168.2.100:8181/apidoc/explorer/index.html>

V tomto prostredí sa nachádza zoznam všetkých dostupných RESTful rozhraní, ktoré sa v kontroléri nachádzajú. Každé z rozhraní ďalej obsahuje strom podrozhraní s možnými HTTP metódami, ktoré sa dajú na nich vykonávať (väčšinou podmnožinu s množinou metód *GET*, *PUT*, *POST* a *DELETE*). Po navigácii do podrozhrania využívajúceho konkrétnu HTTP metódu sa zobrazí jeho dokumentácia a dá sa z neho priamo otestovať poslanie danej metódy (po zadanií prípadných parametrov, respektíve tela správy). API Explorer využíva ako prednasta-vený formát JavaScript Object Notation (JSON) pre zobrazenie odozvy kontroléra.

The screenshot shows the 'Controller Resources' tab selected in the navigation bar. Below the header, a message states: 'Below are the list of APIs supported by the Controller.' A table lists various APIs with their last update date and a row of operations (Show/Hide, List Operations, Expand Operations, Raw). The 'network-topology' API is expanded, showing its resources and methods:

Method	URI	Operations
POST	/config/	Show/Hide List Operations Expand Operations Raw
GET	/config/network-topology:network-topology/	Show/Hide List Operations Expand Operations Raw
PUT	/config/network-topology:network-topology/	Show/Hide List Operations Expand Operations Raw
DELETE	/config/network-topology:network-topology/	Show/Hide List Operations Expand Operations Raw
POST	/config/network-topology:network-topology/	Show/Hide List Operations Expand Operations Raw
GET	/config/network-topology:network-topology/topology/{topology-id}/	Show/Hide List Operations Expand Operations Raw
PUT	/config/network-topology:network-topology/topology/{topology-id}/	Show/Hide List Operations Expand Operations Raw

Obr. 22: API Explorer

7 Záver

Cieľom tejto práce bolo uviesť problematiku SDN sietí, opísť jej princípy, architektúru a protokoly, ktoré sa v nej používajú (so zámerom na protokol OpenFlow). Na základe týchto poznatkov sa malo navrhnuť a zrealizovať zapojenie tejto siete na katedre KIS, pričom sa tak malo vykonať v prostredí s reálnymi sieťovými zariadeniami, ako aj v emulovanom prostredí programu Mininet. Preskúmať sa mali aj možnosti rozšírenia funkcionality takejto siete prostredníctvom programovania. Príklad programového rozšírenia kontroléra bol po dohode s vedúcim práce vynechaný, nakoľko by si vyžadoval viac času a priestoru.

Myslím si, že som všetky vytýčené ciele tejto práce splnil. Kedže technológie SDN sietí sa len nedávno začali rýchlosťou vyvíjať, je to stále chaotické prostredie, ktoré ešte potrebuje určité formy štandardizácie. Aj tieto faktory mi stažili písanie práce, kedže väčšina technológií bola pre mňa neznámych a v niektorých častiach neboli dostatok dobrej dokumentácie, respektívne existovalo viaceru odlišných definícií a pohľadov na tú istú problematiku.

Vo svojej práci som narazil na viaceru aspektov SDN sietí, ktoré by sa dali hlbšie preskúmať. Ku protokolu OpenFlow existuje niekoľko spomenutých alternatív, ako riešiť komunikáciu medzi SDN kontrolérom a sieťovými zariadeniami. Taktiež neboli čas hlbšie preskúmať všetky funkcionality kontroléra OpenDaylight, ktorý napríklad umožňuje prepojenie s clouдовým riešením OpenStack, ktorý bol momentálne v rámci diplomových prác takisto riešený a nasadený na katedre KIS. Rovnako zaujímavá je možnosť rozšírenia funkcií kontroléra programovaním, či už prostredníctvom vytvárania modulov v rámci kontroléra, alebo vytváraním samostatných aplikácií komunikujúcich s kontrolérom cez RESTful rozhrania a iné. V práci sa nenašlo dostatok priestoru ani na lepšie preskúmanie ostatných SDN kontrolérov.

Pre výučbu SDN sietí na katedre KIS by som navrhoval použiť emulátor Mininet, v ktorom sa dá jednoducho a rýchlo vytvárať virtuálna sieť. Emulátor navyše štandardne používa softvérový prepínač Open vSwitch, ktorý sa reálne používa vo virtualizovaných prostrediaciach firiem a tiež ako OpenFlow agent pre fyzické prepínače. Ako fyzické prepínače sa dajú použiť aj prispôsobené vstavané počítače a v rámci finančných možností aj spomínaný štandardný prepínací hardvér. Ako SDN kontrolér je OpenDaylight pravdepodobne najrozšíahlejší. Vhodný je aj pre výučbu, pričom by sa poprípade mohol na jej počiatku použiť aj jednoduchší SDN kontrolér, pre názornejšiu ukážku fungovania SDN. Na interne sa nachádza aj niekoľko predoma pripravených VM obrazov, ktoré obsahujú programy ako Mininet a viaceré SDN kontroly. Sú určené pre štúdium SDN sietí a boli by preto vhodné pre ich výučbu na katedre KIS.

Kedžže sa SDN siete začali vo svete presadzovať len nedávno a nie je ani pevne dohodnuté, čo presne sa dá za SDN siet označiť, neexistuje ešte na túto problematiku veľa vhodných, celistvých študijných materiálov. V roku 2013 bol na Princetonovej univerzite v USA otvorený prvý SDN kurz, ktorý sa aj nadálej aktívne rozvíja a je voľne dostupný prostredníctvom stránky [coursera.org](https://www.coursera.org). Spôsob výučby a študijné materiály v ňom použité by taktiež mohli pomôcť pri vytváraní materiálov pre výučbu na katedre KIS.

SDN siete sú rapídne sa rozvíjajúcou technológiou s veľkým potenciálom. Predpokladám, že ich význam bude postupom času len narastať. Existovalo už viacero sľubných technológií, ktoré však kvôli nezáujmu spoločnosti, nutnosti rozsiahlych zmien v pôvodnej sieti, alebo nedostatku okamžitých výhod oproti ustáleným riešeniam zanikli. Hlavný predstaviteľ SDN sietí, protokol OpenFlow, si však nevyžaduje veľké úpravy v sietových zariadeniach, pretože využíva už existujúce prvky týchto zariadení, čo zvýšilo popularitu SDN technológie. SDN si takisto už našlo skupinu používateľov (hlavne dátové centrá), ktorým poskytuje okamžité výhody oproti zaužívaným riešeniam a odstraňuje veľké problémy, ktoré už začínali robiť správu niektorých sietí nezvládnuteľnou. Takéto skupiny používateľov tvoria pevný základ pre rozvoj SDN technológie, ktorá začína nachádzať možné uplatnenia aj v iných oblastiach, ako napríklad v prostredí WAN sietí, bezdrôtových sietí, v sietovej bezpečnosti a ďalších. Človek, ktorý sa chce venovať počítačovým sietiam by sa preto mal s technológiou SDN sietí oboznámiť.

Literatúra

- [1] Open Networking Foundation. *Software-Defined Networking: The New Norm for Networks*. 2012. URL: <https://www.opennetworking.org/images/stories/downloads/sdn-resources/white-papers/wp-sdn-newnorm.pdf>.
- [2] Scott Shenker. *Software-Defined Networking at the Crossroads*. 2013. URL: <https://www.youtube.com/watch?v=WabdXYzCAOU>.
- [3] Thomas D. Nadeau and Ken Gray. *SDN: Software Defined Networks*. O'Reilly Media, Inc., 2013. ISBN: 978-1-4493-4230-2. URL: <http://it-ebooks.info/book/2888/>.
- [4] Pate Pryson. *NFV and SDN: What's the Difference?* 2013. URL: <https://www.sdxcentral.com/articles/contributed/nfv-and-sdn-whats-the-difference/2013/03/>.
- [5] Rivka G. Little. *Top SDN news and trends of 2014*. 2014. URL: <http://searchsdn.techtarget.com/news/2240237374/Top-SDN-news-and-trends-of-2014>.
- [6] *SDN use cases*. URL: <https://www.sdxcentral.com/sdn-use-cases/>.
- [7] Jon Oltsik. *SDN uses for network security*. 2014. URL: <http://www.networkworld.com/article/2598056/cisco-subnet/enterprise-security-professionals-speak-out-on-sdn-use-cases-for-network-security.html>.
- [8] SDxCentral. *Who is the Open Networking Foundation (ONF)?* URL: <https://www.sdxcentral.com/resources/sdn/who-is-open-networking-foundation-onf/>.
- [9] Open Networking Foundation. *OpenFlow Switch Specification 1.3.3*. Version 1.3.3. 2013. URL: <https://www.opennetworking.org/images/stories/downloads/sdn-resources/onf-specifications/openflow/openflow-spec-v1.3.3.pdf>.
- [10] Open Networking Foundation. *OpenFlow Switch Specification 1.5.0*. Version 1.5.0. 2014. URL: <https://www.opennetworking.org/images/stories/downloads/sdn-resources/onf-specifications/openflow/openflow-switch-v1.5.0.noipr.pdf>.
- [11] Cubieboard Docs. URL: <http://docs.cubieboard.org/products/start>.
- [12] Cubieboard1. URL: http://docs.cubieboard.org/_detail/cubieboard.jpg?id=products%3Astart.
- [13] Cubieboard2. URL: http://docs.cubieboard.org/_detail/a20-cubieboard.png?id=products%3Astart.

- [14] *Banana Pi Specification*. URL: <http://www.bananapi.org/p/product.html>.
- [15] *Banana Pi*. URL: http://cdn-reichelt.de/bilder/web/xxl_ws/A300/BANANA_PI_01.png.
- [16] *Raspberry Pi Specification*. URL: <http://www.raspberrypi.org/products/model-b-plus/>.
- [17] *Raspberry Pi*. URL: <http://www.kiwi-electronics.nl/image/cache/data/products/raspberry-pi/rasppi-mod-b-1-800x533.jpg>.
- [18] *NetFPGA Guide*. URL: <https://github.com/NetFPGA/netfpga/wiki/Guide>.
- [19] Greg Ferro. *Blessay: Comparing Merchant and Custom Silicon*. URL: <http://etherealmind.com/analysis-merchant-custom-silicon/>.
- [20] Rob Sherwood. *Tutorial: White Box/Bare Metal Switches*. URL: <http://www.bigswitch.com/sites/default/files/presentations/onug-baremetal-2014-final.pdf>.
- [21] William Choe. *Bare Metal Networking, Then and Now...* URL: <http://cumulusnetworks.com/blog/bare-metal-networking/>.
- [22] *What is Cisco OpenFlow?* URL: <https://www.sdxcentral.com/resources/cisco/cisco-openflow/>.
- [23] *Junos OS OpenFlow Feature Guide*. URL: <http://www.juniper.net/techpubs/en-US/junos14.2/information-products/pathway-pages/junos-sdn/junos-sdn-openflow.pdf>.
- [24] *EZchip NPS Family of Network Processors*. URL: <https://www.sdxcentral.com/products/nps-400/>.
- [25] *Indigo Virtual Switch*. URL: <http://www.projectfloodlight.org/indigo-virtual-switch/>.
- [26] *Open vSwitch*. URL: <https://github.com/openvswitch/ovs>.
- [27] *Project Floodlight*. URL: <http://www.projectfloodlight.org/floodlight/>.
- [28] *Ryu SDN framework*. URL: <http://osrg.github.io/ryu/>.
- [29] *ONOS whitepaper*. URL: <http://onosproject.org/wp-content/uploads/2014/11/Whitepaper-ONOS-final.pdf>.
- [30] *OpenDaylight FAQ*. URL: <http://www.opendaylight.org/project/faq>.

- [31] *OpenDaylight TECHNICAL OVERVIEW*. URL: <http://www.opendaylight.org/project/technical-overview>.
- [32] *HP VAN SDN Controller*. URL: <http://h20195.www2.hp.com/v2/getpdf.aspx/4AA4-9827ENW.pdf>.
- [33] *Mininet Overview*. URL: <http://mininet.org/overview/>.
- [34] *OpenDaylight User Guide*. URL: <https://www.opendaylight.org/sites/opendaylight/files/User-Guide-Helium-SR2.pdf>.
- [35] Brent Salisbury. *OpenFlow: Proactive vs Reactive Flows*. 2013. URL: <http://networkstatic.net/openflow-proactive-vs-reactive-flows/>.
- [36] *Postman Documentation*. URL: <http://www.getpostman.com/docs>.
- [37] *Controller Modules/Bundles and Interfaces*. 2013. URL: https://wiki.opendaylight.org/view/Controller_Projects%27_Modules/Bundles_and_Interfaces.
- [38] *OpenDaylight Controller: Architectural Framework*. 2013. URL: https://wiki.opendaylight.org/view/OpenDaylight_Controller:Architectural_Framework.
- [39] *OpenDaylight MD-SAL Restconf API Explorer*. 2015. URL: https://wiki.opendaylight.org/view/OpenDaylight_Controller:MD-SAL:Restconf_API_Explorer.

Príloha A: tree_topo.py

```
#!/usr/bin/python

from mininet.net import Mininet
from mininet.node import RemoteController
from mininet.node import CPULimitedHost, Host, Node
from mininet.node import OVSKernelSwitch
from mininet.cli import CLI
from mininet.log import setLogLevel, info
from mininet.link import TCLink, Intf
from subprocess import call

def myNetwork():

    #definovanie siete
    net = Mininet( topo=None,
                   build=False,
                   ipBase='192.168.2.0/24')

    #nastavenie parametrov pre pripojenie na kontroler
    info('*** Adding controller\n')
    c0 = net.addController(name='c0',
                           controller=RemoteController,
                           ip='192.168.2.100',
                           protocol='tcp',
                           port=6653)

    #vytvorenie Open vSwitch prepinacov
    info('*** Add switches\n')
    s1 = net.addSwitch('s1', cls=OVSKernelSwitch, protocols='OpenFlow13')
    s2 = net.addSwitch('s2', cls=OVSKernelSwitch, protocols='OpenFlow13')
    s3 = net.addSwitch('s3', cls=OVSKernelSwitch, protocols='OpenFlow13')

    #vytvorenie koncovych stanic
    info('*** Add hosts\n')
    h1 = net.addHost('h1', cls=Host, mac='00:00:00:00:00:01',
                     ip='192.168.2.11/24')
    h2 = net.addHost('h2', cls=Host, mac='00:00:00:00:00:02',
                     ip='192.168.2.12/24')
    h3 = net.addHost('h3', cls=Host, mac='00:00:00:00:00:03',
                     ip='192.168.2.13/24')
    h4 = net.addHost('h4', cls=Host, mac='00:00:00:00:00:04',
                     ip='192.168.2.14/24')

    #pridanie liniek medzi prepinacmi a koncovymi stanicami
    #Gigabitovy prepoj medzi prepinacmi
    #Fastethernet na pripojenie koncovych stanic
    info('*** Add links\n')
```

```

s1s2 = {'bw':1000}
net.addLink(s1, s2, cls=TCLink , **s1s2)
s1s3 = {'bw':1000}
net.addLink(s1, s3, cls=TCLink , **s1s3)
s2h1 = {'bw':100}
net.addLink(s2, h1, cls=TCLink , **s2h1)
s2h2 = {'bw':100}
net.addLink(s2, h2, cls=TCLink , **s2h2)
s3h3 = {'bw':100}
net.addLink(s3, h3, cls=TCLink , **s3h3)
s3h4 = {'bw':100}
net.addLink(s3, h4, cls=TCLink , **s3h4)

#vybudovanie siete
info('*** Starting network\n')
net.build()

#pripojenie prepinacov na kontroler
info('*** Starting switches\n')
net.get('s3').start([c0])
net.get('s2').start([c0])
net.get('s1').start([c0])

#spristupnenie CLI emulátora
info('*** Post configure switches and hosts\n')
CLI(net)
net.stop()

if __name__ == '__main__':
    setLogLevel('info')
    myNetwork()

```