UPRAVLJANJE PROMETOM U L2 SDN MREŽI

Posarić, Dario

VISOKO UČILIŠTE ALGEBRA

ZAVRŠNI RAD

UPRAVLJANJE PROMETOM U L2 SDN MREŽI

Dario Posarić

Zagreb, rujan 2018.

Student vlastoručno potpisuje Završni rad na prvoj stranici ispred Predgovora s datumom i oznakom mjesta završetka rada te naznakom:

„Pod punom odgovornošću pismeno potvrđujem da je ovo moj autorski rad čiji niti jedan dio nije nastao kopiranjem ili plagiranjem tuđeg sadržaja. Prilikom izrade rada koristio sam tuđe materijale navedene u popisu literature, ali nisam kopirao niti jedan njihov dio, osim citata za koje sam naveo autora i izvor, te ih jasno označio znakovima navodnika. U slučaju da se u bilo kojem trenutku dokaže suprotno, spreman sam snositi sve posljedice uključivo i poništenje javne isprave stečene dijelom i na temelju ovoga rada“.

U Zagrebu, 4. 9. 2018.

Dario Posavić

Predgovor

Posebno bih zahvalio profesoru Silviju Papiću koji me je svojim mentorstvom, pristupom rješavanju problema te voljom, predanošću i poticanjem motivirao za izvrsnost od samih početaka naše suradnje.

Želim zahvaliti svim profesorima i asistentima Visokog učilišta Algebra koji su mi u protekle tri godine prenijeli znanja i vještine iz polja sistemskog inženjerstva.

Zahvalio bih svim kolegama i prijateljima te posebno svojoj obitelji koja mi je pružala najveću podršku tijekom studija.

Prilikom uvezivanja rada, Umjesto ove stranice ne zaboravite umetnuti original potvrde o prihvaćanju teme završnog rada kojeg ste preuzeli u studentskoj referadi

Sažetak

Softverski definirane mreže (engl. Software Defined Networking – SDN) nova su paradigma u svijetu računarstva kojom se mijenja način implementacije i upravljanja nad mrežom i mrežnim uređajima. Temeljne dvije karakteristike SDN-a su odvajanje kontrolne i podatkovne ravni te uspostavljanje centralnog mjesta kontrole nad cjelokupnom mrežom u vidu SDN kontrolera. Kontroler kao centralno mjesto u mreži nudi sučelje za pristup aplikacijama koje iste mogu koristiti za različite svrhe kao što su upravljanje prometom, upravljanje sigurnosnim politikama u mreži, optimizacijom mreže i dr. U ovom radu prikazana je funkcionalnost SDN aplikacije napisane od strane autora kojom se vrši upravljanje prometom unutar SDN topologije. Prilikom prikaza rada aplikacije izvršena su mjerenja vremena konvergencije u usporedbi s konvergencijom protokola u tradicionalnoj mreži korištenjem Spanning Tree protokola (STP 802.1d), ostvareno je povoljnije vrijeme dostupnosti mreže, manja latencija te je prikazana mogućnost granularnog pristupa upravljanju prometom u SDN mreži. Prilikom demonstracije implementacije SDN-a i tradicionalne mreže prikazane su bitne prednosti u vremenu dobivanja informacije o trenutačnom stanju mrežne topologije.

Ključne riječi: SDN, Tradicionalna mreža, Openflow protokol, granularno upravljanje prometom, konvergencija mreže.

Abstract

Software-defined networks are a new paradigm in the world of computing which significantly changes the way in which we deploy and manage our networks and devices. The basic two features of SDN are the separation of control and data planes and the establishment of a central control site in the form of SDN controller. The controller as a central place in the network provides an interface to network applications that can be used for different purposes such as traffic management, network security management, network optimization, etc. This paper presents the functionality of a SDN application written by the author of this thesis for the purpose of traffic management within the SDN topology. After testing and running the application we marked the difference in convergence time between SDN and traditional networks that use STP and RSTP protocols, we observed better time of network availability and lower latency in SDN network, we also demonstrated how granular access to traffic management in the SDN network can be achieved. During a demonstration of SDN and traditional network implementation we demonstrated significant advantages in times that are needed to obtain an information about the current state of network topology.

Ključne riječi: SDN, Traditional network, Openflow protocol, Granular traffic engineering, Network convergence

Sadržaj

1. Uvod

Umrežavanje i mrežne tehnologije doživjele su mnogo promjena u svojoj dugoj evoluciji, no način na koji upravljamo temeljnim elementima mrežnog sustava kao što su usmjernici, preklopnici i razni *middleware* uređaji ostali su unatrag nekoliko desetljeća isti. Dok druge sfere računarstva neumoljivo koračaju naprijed u sferi automatizacije, orkestracije te centralnog nadzora i kontrole, mreže su dalje po svojoj prirodi vrlo distribuirane te vertikalno integrirane, što u biti znači veću kompleksnost i težu upravljivost. Cilj je promjena paradigme u vidu softverski definiranih mreža promijeniti taj trend odvajanjem upravljačke od podatkovne ravni te osigurati programibilnu, skalabilnu i fleksibilnu mrežu spremnu na zahtjeve budućnosti.

U radu će biti obrađeni tehničko-tehnološki koncepti softverski definiranih mreža uz usporedbu s tradicionalnom mrežom u vidu arhitekture, implementacije upravljačke i podatkovne ravni, instalacije i konfiguracije, upravljanja i održavanja te troškova i koristi za male i srednje velike organizacije.

Naglasak praktičnog dijela bit će stavljen na usporedbu softverski definirane te tradicionalne mreže, i to u vidu upravljanja prometom u mreži te efikasnosti dobivanja globalnog pogleda na topologiju. Za omogućavanje funkcionalnosti upravljanja prometom u okolini SDN mreže bit će prikazan rad interaktivne *bash* skripte napisane od strane autora koja će preko JSON (Javascript Object notation)[1] poziva vršiti autentikaciju na SDN kontroler, dohvatiti podatke o uređajima u mreži te vršiti mogućnost specificiranja toka podataka između izvorišta i odredišta. Za potrebe usporedbe SDN rješenja s upravljanjem prometom u okolini tradicionalne topologije vršit će se manipulacijom STP protokola te Rapid Spanning tree protokola (RSTP, 802.1w standard).

Kao glavni pokazatelj uspješnosti funkcionalnog upravljanja prometom mjerit će se vrijeme nedostupnosti mreže, kao i broj odbačenih paketa prilikom njezina izvršenja, a tako će se prikupiti podaci na temelju kojih će se izvršiti analiza te usporedba jedne i druge opcije upravljanja prometom.

[1] JSON – otvoreni standard za razmjenu tekstualnih podataka

2. Osnovne karakteristike i koncepti softverski definiranih mreža

Živimo u suvremenom informacijskom dobu kada su kreacija, distribucija, uporaba, integracija i manipulacija informacijama postale važan faktor ekonomskih, političkih i kulturnih aktivnosti ljudskog društva, a iza svega toga stoje informacijske tehnologije.

Transformacija društva u novom digitalnom dobu teži povezivanju svih fizičkih uređaja na Internet, u pozadini čega ključnu važnost imaju mrežne tehnologije. Osim toga, u poslovnim okolinama mrežne tehnologije postale su srž distribucije informacija unutar samog poslovnog informacijskog sustava. Napretkom virtualizacijskih tehnologija, kao i dostupnosti nam sve većeg kapaciteta memorije, procesorske snage i prostora za pohranu podataka zakoračili smo u doba tzv. elastičnog računarstva. U takvim uvjetima mreža kao okosnica protoka informacija u računalnim sustavima doživljava svoju transformaciju iz tradicionalne u softverski definiranu.

S tehničko-tehnološkog aspekta činjenica je da trenutačne informacijske i komunikacijske tehnologije koje omogućavaju tu povezanost i sveopću upotrebu postaju u svojoj srži sve kompleksnije i teže za upravljanje. To vrijedi jer tradicionalne mreže vertikalno su integrirane. Kontrolna i podatkovna ravan implementirane su zajedno na mrežnim uređajima kao što su preklopnici i usmjernici, što znači da svaki od tih uređaja ima vlastitu „pamet“ i da svaki od njih radi vlastite odluke o usmjeravanju paketa ili okvira ovisno o logici koja je definirana pomoću usmjerničkih protokola kao što su Border Gateway Protocol (BGP), Open Shortest Path First (OSPF) i Enhanced Interior Gateway Routing Protocol (EIGRP), pomoću statičkih ruta ili temeljeno na zapisima koji se nalaze u Content-addressable memory (CAM)[2] tablicama preklopnika.

Kontrolna ravan odlučuje što napraviti s podatkom, dok podatkovna ravan radi konkretnu aktivnost slanja, tj. prosljeđivanja podatka. Svaki od tih elemenata nalazi se distribuiran na mrežnim uređajima.

Takav dizajn mrežnih uređaja razvijan je upravo zbog evolucije Interneta kao medija gdje su ovakve karakteristike mreže bile poželjne i namjerno tako dizajnirane kroz povijest. U

[2] Content-addressable memory – posebna vrsta memorije korištena za brza i specifična pretraživanja

slučaju prekida rada manjeg ili većeg dijela mreže ostali uređaji imaju „pamet“ da usmjere promet prema krajnjoj destinaciji, ali drugim smjerom.

Ovakav pristup i koncept usmjeravanja i preklapanja nosi sa sobom visoke tehničko-tehnološke zahtjeve u vidu administracije. Administratori trebaju konfigurirati svaki uređaj zasebno koristeći različite komande niže razine koje definiraju proizvođači.[3]

Takvi tradicionalni uređaji rade na specijaliziranom i često vrlo skupom sklopovlju, što sa sobom nosi i visoke troškove.

Ipak, najveći problem takvom pristupu umrežavanja nosi činjenica da je tradicionalnu mrežu teško programirati, orkestrirati i automatizirati. Ručni rad je neizbježan, kao i posebno konfiguriranje svakog uređaja u mreži zasebno.

Ideja i koncept softverski definiranih mreža teži promijeniti tradicionalni pristup umrežavanju odvajanjem kontrolne ravni od podatkovne ravni te centraliziranje kontrolne ravni u vidu logički centraliziranog kontrolera odnosno mrežnog operativnog sustava koji definira što ostali uređaji u mreži trebaju napraviti s prometom. Važno je napomenuti da takav koncept ne znači nužno i fizički centraliziran sustav, već u većini slučajeva distribuiran sustav mrežnih kontrolera.[4]

Mrežnom kontroleru primarna je zadaća da bude medijator između same mreže te mrežnih aplikacija, što znači da mora postojati separacija komunikacije prema mreži i prema aplikacijama. Nekadašnja upravljačka logika koja je bila smještena u svakom uređaju zasebno sada je smještena samo u centralnom kontroleru i on ima kompletnu kontrolu nad stanjima u podatkovnom dijelu mreže, tu kontrolu ostvaruje protokolima „*Southbound“* sučelja, od kojih je najpoznatiji Openflow protokol. S druge strane, imajući sliku i kontrolu nad čitavom mrežom, kontroler može aplikacijama dati na korištenje globalni i apstrakcijski prikaz same fizičke ili logičke mreže. Ta komunikacija ostvaruje se „*Northbound*“ protokolima od kojih je najčešće korišteni REpresentational State Transfer (RESTful)[5] protokol. RESTful protokol, za razliku od Openflow protokola, još uvijek nije potpuno standardiziran, mrežne aplikacije mogu koristiti to sučelje za dobivanje prikaza i stanja mreže, ali i za direktno djelovanje nad samom mrežom.

[3] Software-defined Networking: A comprehensive Survey, str. 2
[4] Ibidem
[5] REST – arhitekturni stil pristupa web servisima

Kao što vidimo, koncept softverski definiranih mreža teži upravo ka pojednostavnjenju upravljanja mrežom povećanjem njezine skalabilnosti, efektivnosti, fleksibilnosti i efikasnosti tako da odvaja upravljačku ravan, koja odlučuje što napraviti s prometom, od podatkovne ravni, koja konkretno vrši slanje prometa. Pojednostavnjuje mrežu konsolidacijom upravljačke ravni tako da definira jedan ili više kontrolera koji upravljaju većim skupom uređaja koji vrše slanje podataka, i to upotrebom unaprijed definiranog sučelja.[6]

2.1. Razvoj i uloga SDN-a danas i u budućnosti

Prema riječima Rohita Mehre, potpredsjednika mrežnih operacija u IDC-u, sklopovlje je definiralo mrežne tehnologije posljednjih 30 godina, a u budućnosti softver će preuzeti glavnu paradigmu.[7]

Cijena upravljanja trenutačnim mrežnim tehnologijama jednostavno postaje prevelika i u sporom kasu prati korak s promjenama u ostalim sferama računarstva. Ono što nas očekuje jest potpuno novi pogled na mrežu, onaj u kojem je softver glavni pokretač promjena.[8]

Povijest softverski definiranih mreža započinje sredinom 90-ih godina, u doba kada je Internet kao medij ušao u široku uporabu. Internet kao mreža svih mreža prati svoje početke u vojnim i akademskim krugovima i kao takav nije nikada zamišljen za uporabu izvan tih sfera. Sa zahtjevima za uporabom sve raznovrsnijih servisa i aplikacija u znanstvenim krugovima započet je niz istraživanja u vidu programibilnih mreža kako bi se unaprijedila fleksibilnost, skalabilnost i efikasnost same mreže, a samim time i poboljšalo korisničko iskustvo.

Aktivne mreže predstavljaju rani oblik uporabe koncepta programibilnih mreža. U ono doba ovakav koncept predlagao je promjene u radu mrežnih uređaja, aktivne mreže omogućavale su slanje programa zajedno s paketima podataka. Ti programi pokretali su se na mrežnim uređajima i tako aktivno utjecali na ponašanje same mreže. Takav koncept nikada nije zaživio izvan akademske zajednice zbog brige oko eventualnih sigurnosnih propusta i problema s performansama.[9]

---

[6] The Road to SDN, str.1
[7] https://www.networkworld.com/article/3046457/data-center/the-networked-world.html, 24.07.2018
[8] Ibidem
[9] The Road to SDN, str. 4

Kralj stoljeća obilježilo je povećanje količine podataka na Internetu, a u isto vrijeme i sve veće potrebe za pouzdanošću uvjetovane od poslovnih i privatnih korisnika. Kako bi se moglo utjecati na kvalitetu usluge, nužno je bilo razvijati razne mehanizme kontrole puta. Nove izazove nosila je i činjenica da tradicionalni usmjernički protokoli nikada nisu razvijani primarno s ciljem upravljanja tokom podataka. Sama činjenica da su kontrolna i podatkovna ravan integrirane unutar svakog od mrežnih uređaja značila je i poteškoće u implementaciji i otklanjanju problema, ali i u predviđanju putanje toka podataka. Bilo je jasno da bi se odvajanjem kontrolne i podatkovne ravni te smještanje kontrolne ravni na zaseban uređaj otklonili mnogi problemi te pojednostavnila konfiguracija.

Razni pružatelji telekomunikacijskih usluga radili su aktivno s istraživačkom zajednicom na razvoju čitavog niza protokola, od kojih vrijedi izdvojiti protokole kao što su ForCES i 4D.

FORces je bio prvi protokol koji je definirao jasnu granicu između kontrolne i podatkovne ravni. Unutar ForCES protokola definirana su dva logička elementa, upravljački element i kontrolni element. Temeljna ideja bila je separacijom ova dva entiteta omogućiti svakom od njih da se razvija neovisno jedan o drugom, s ciljem ubrzanja razvoja svakoga. [10]

4D projekt je, s druge strane, definirao arhitekturu s četiri sloja. Sloj odlučivanja (engl. Decision layer), koji je odgovoran za izradu konfiguracije, diseminacijski sloj (engl. Dissemination layer), odgovoran za pružanje informacije o mreži sloju odlučivanja, Otkrivajući sloj (engl. Discovery layer), odgovoran za otkrivanje mrežne topologije, i podatkovni sloj (engl. Data plane), odgovoran za slanje podataka. [11]

U drugoj polovici desetljeća daljnjem razvoju projekta 4D podizanjem sigurnosti čitavog sustava u vidu projekta SANE/Ethane na Sveučilištima Stanford i Berkeley otvorena su vrata razvoju danas dominantnog SDN protokola, Openflow.

Openflow je nastao 2008. godine u akademskim krugovima s ciljem kreacije platforme koja bi služila za eksperimentiranje s otvorenim mrežnim tehnologijama u sveučilišnim kampus mrežama. No nedugo nakon što je izdana prva verzija protokola, daljnjim razvojem i uvidom u prednosti i mogućnosti koje ova tehnologija ima, osnovana je i neprofitna organizacija Open Network Foundation (ONF) u čijem se upravljačkom vijeću nalazi velik broj pružatelja usluga kao što su Google, Microsoft i Facebook. Glavna zadaća OFN-a je promoviranje,

[10] Software Defined Networking Concepts, str. 9
[11] Ibidem

razvoj i komercijalizacija SDN-a, a upravo su te organizacije među prvima implementirale Openflow unutar svojih produkcijskih okolina. Slično kao i u slučaju Ipv6 protokola, Openflow je dizajniran da može raditi s drugim mrežnim tehnologijama i protokolima simultano, što mu je velika prednost kod budućih migracija koje za korisnike moraju proći neprimjetno i bez prekida u radu. Naime, mnogi današnji preklopnici podržavaju hibridni način rada u kojem je u isto vrijeme moguće koristiti Openflow komunikaciju, kao i komunikaciju koja je bazirana na tradicionalnim mrežnim protokolima.

Gorannson i Black definiraju pet otvorenih karakteristika koje neka mrežna tehnologija mora imati kako bi se mogla smatrati SDN-om, a to su:

- separacija slojeva
- jednostavan uređaj na podatkovnom sloju
- središnja kontrola
- automatizacija i virtualizacija mreže
- otvoreni standardi. [12]

Openflow protokol jedini je protokol koji udovoljava svim navedenim karakteristikama te se smatra začetnikom SDN-a. Zapravo, prije Openflowa termin SDN nije ni postojao. Standard je to koji je su prihvatili i akademska zajednica i industrija koja ga u današnje vrijeme sve više podržava u uređajima koje proizvodi.

Google je prvi među velikim korporativnim divovima još 2012. godine kompletnu *backbone* mrežu migrirao na Openflow. To im je omogućilo da razviju i implementiraju vlastite aplikacije za upravljanje zagušenjem u prometu, da centraliziraju kontrolu nad svojom kompletnom *Wide Area Network (WAN)*[13] infrastrukturom te da sve preklopnike u mreži zamijene interno razvijenim sklopovljem, što im je omogućilo visoke uštede. Google je ustvrdio da je upotrebom Openflow protokola uspio ostvariti 100 % iskoristivosti svih svojih *backbone* linkova. Prije migracije te su se brojke kretale oko 40 %.[14]

Unatoč prednostima koje nose softverski definirane mreže, trenutačno je stanje u industriji takvo da se osim u okruženjima velikih organizacija kao što je Google, SDN i dalje ne uvodi

[12] Software Defined Networks, poglavlje 3.
[13] WAN: Podatkovna mreža koja pokriva veće geografsko područje
[14] https://www.wired.com/2012/04/going-with-the-flow-google/, 27.07.2018

ubrzanim korakom kao što bi se to moglo očekivati. Postoje razlozi zbog kojih većina organizacija odgađa implementaciju SDN-a, a to su prije svega:

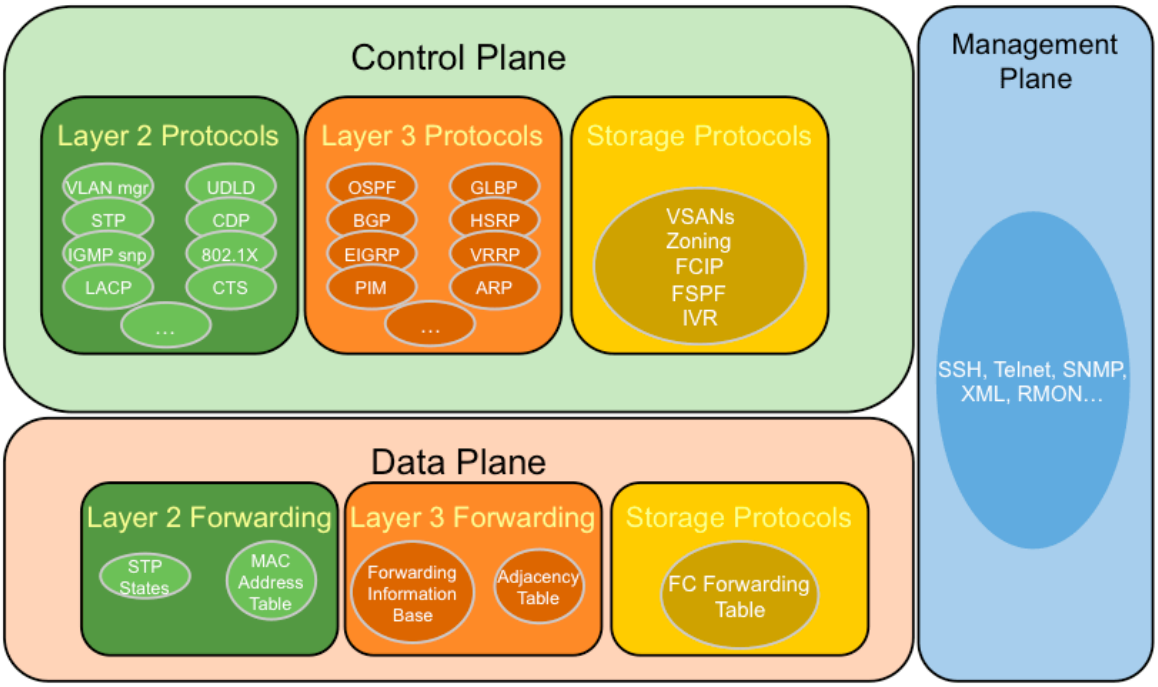
- konzervativno stajalište prema inovativnim tehnologijama
- brige oko nezrelosti i nestabilnosti proizvoda
- nedostatak pokretača unutar poslovnih procesa
- vezanost za tehnologiju i proizvode određenih proizvođača
- nedostatak znanja i vještina kod implementacije te naknadnog održavanja
- vezanost za tehnologiju određenog proizvođača opreme[15].

[15] https://blogs.gartner.com/andrew-lerner/2017/08/29/the-state-of-sdn-september-2017/, 26.07.2018

3. Usporedba tradicionalne mreže i SDN mreže

3.1. Arhitektura

Arhitekturu preklopnika i usmjernika u tradicionalnim mrežama obilježava distribuirana narav upravljačke, kontrolne i podatkovne ravni.



Slika 3.1 Arhitektura tradicionalnih mrežnih uređaja[16]

Na upravljačkom sloju tradicionalnog mrežnog uređaja nalaze se protokoli kojima se vrši pristup i konfiguriranje samog uređaja, protokoli kao što su SSH, Telnet, HTTP i drugi. Kada administratori pristupaju uređaju kako bi unijeli specifične komande i konfiguracije, pristupaju joj na upravljačkom sloju, a te komande se bilježe, spremaju i šalju na daljnje procesuiranje kontrolnom sloju.

Na kontrolni sloj možemo gledati kao na mozak mrežnog uređaja. Tu nalazimo L3 protokole kao što su OSPF, EIGRP, BGP i druge te L2 protokole kao što su STP, Cisco Discovery Protocol (CDP) i Link Aggregation Control Protocol (LACP). Kontrolni sloj svakog uređaja direktno komunicira s kontrolnim slojevima drugih susjednih uređaja te na temelju

[16]https://www.globalknowledge.com/us-en/content/articles/how-to-secure-cisco-routers-and-switches, 26.07.2018

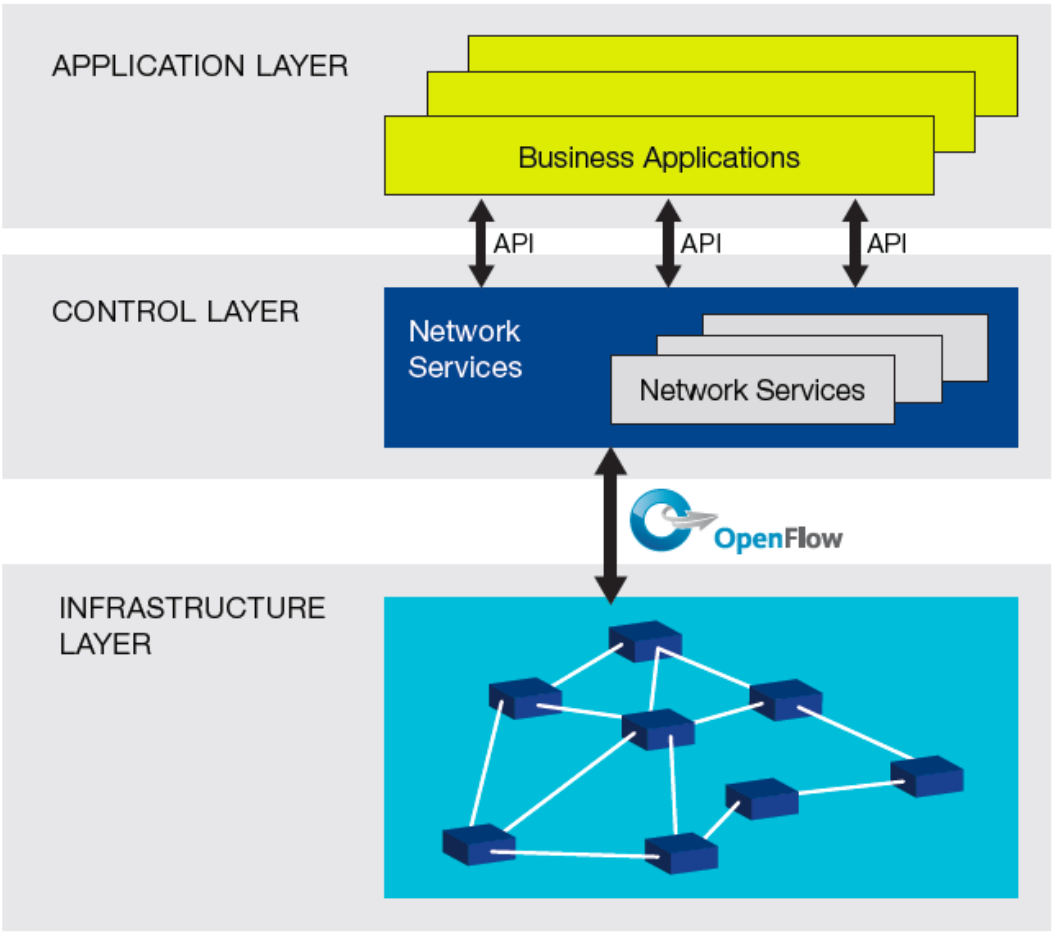
razmijenjenih informacija određuje što će napraviti s prometom, a prosljeđivanje prometa vrši se konkretno na podatkovnom sloju.

Podatkovni sloj vrši krajnje prosljeđivanje paketa sukladno pravilima koja je definirao kontrolni sloj.

S druge strane, SDN okvir sastoji se od tri temeljna sloja:

- **Infrastrukturni sloj** – primarna zadaća infrastrukturnog sloja je usmjeravanje podataka u mreži. Sastoji se od tzv. *whitebox preklopnika*[17] koji implementiraju minimalan set softvera pogodan za SDN okolinu na uobičajenom i jeftinom sklopovlju
- **Kontrolni sloj** – primarna zadaća je programiranje i kontrola nad infrastrukturnim slojem, a to se postiže popunjavanjem *flow* tablica elemenata infrastrukturnog sloja; u centru kontrolnog sloja nalazi se mrežni operacijski sustav, tj. kontroler
- **Aplikacijski sloj** – sastoji se od samih mrežnih aplikacija koje uz pomoć visoko apstraktnog prikaza mreže mogu direktno utjecati na ponašanje svih uređaja.[18] Aplikacije mogu razvijati proizvođači ili treće strane.

[17] Whitebox preklopnik: Preklopnik s instaliranim minimalnim setom protokola za SDN funkcionalnost
[18] Software-Defined Networking using OpenFlow: Protocols, Applications and Architectural Design Choices, str. 303



Slika 3.2 Arhitektura tipičnog SDN okruženja[19]

Temeljna razlika u arhitekturi između tradicionalne i SDN mreže jest u načinu separacije ovih triju slojeva. U tradicionalnoj mreži svaki uređaj u sebi sadrži implementirane upravljačku, kontrolnu i podatkovnu ravan, a odluku o slanju paketa vrši ovisno o logici koju je kontrolna ravan programirala na podatkovnu ravan. Svaka od tih kalkulacija vrši se distribuirano, drugim riječima, svaki uređaj radi kalkulaciju za sebe, ne postoji centralna kontrola i administracija.

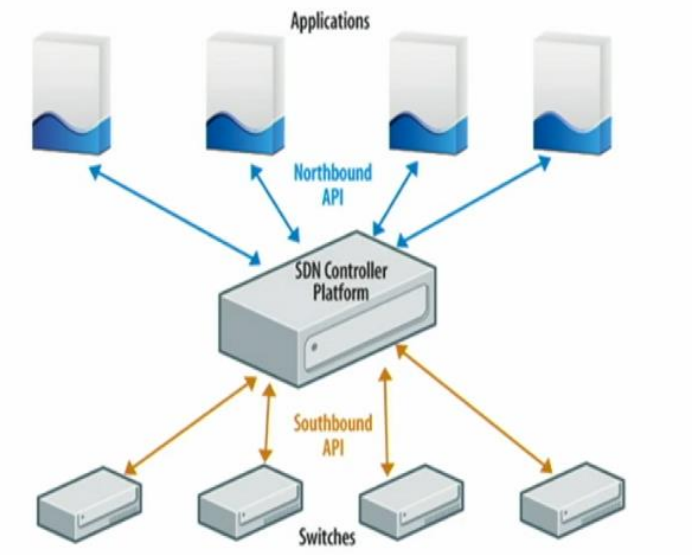
U SDN mreži zadaća svake ravni odvojena je na zasebne uređaje. Preklopnici i usmjernici nalaze se na infrastrukturnom sloju te u sebi imaju implementiranu samo podatkovnu ravan, a sve odluke o tome kuda treba poslati paket određuje kontroler na kontrolnoj ravni i ta pravila zapisuje u tablice preklopnika. Budući da kontrolna ravan ima centralnu kontrolu i pregled nad svim elementima u mreži, ona može ponuditi te informacije na korištenje aplikacijama na zasebnom Aplikacijskom programskom sučelju (API-u). Aplikacije

[19] S. Azodolmolky, Software Defined Networking with Openflow, str. 40

ostvaruju dvosmjernu komunikaciju s kontrolerom na *norhbound* sučelju te tako mogu u realnom vremenu dobivati informacije o mreži te na tu mrežu i utjecati.

3.2. Sučelja i protokoli u SDN okruženju

Ravni u SDN okruženju spajaju *northbound* i *southbound* sučelja te pripadajuće protokole definirane na svakom od njih.



Slika 3.3 Sučelja u SDN mreži[20]

Northbound sučelje nalazi se između kontrolera i SDN aplikacija. Aplikacije putem *northbound* sučelja mogu dohvatiti podatke o svim resursima u mreži te modificirati parametre rada samih uređaja. Najčešće korišteno API *northbound* sučelje je *Representional State Transfer (REST). REST API* počiva na HTTP/S protokolu. Kako bi identificirala određeni resurs u mreži, aplikacija pristupa njegovu URI-ju, tj. jedinstvenom identifikatoru resursa. Zadaća je kontrolera da aplikacijama omogući i osigura pristup svakom od resursa. Aplikacije tada mogu koristiti HTTPS *GET*, *POST*, *PUT* ili *DELETE* pozive kako bi pristupali podacima o resursima u mreži dohvaćanjem informacija o uređajima u JSON formatu.

Southbound sučelje definirano je između infrastrukturnog i kontrolnog sloja SDN mreže, tj. između preklopnika i kontrolera. Trenutačno u industriji postoji čitava lepeza protokola koji se koriste na *southbound* sučelju u različite svrhe. Protokoli kao što su Netconf, *Simple*

[20] J. Casey, INE: Introduction to SDN and Openflow

Network Management Protocol (SNMP) i OPFLEX koriste se za centraliziranu konfiguraciju, administraciju i distribuciju mrežnih politika u mreži. *BGP-LS* je dodatak na *Border Gateway Protokol (BGP)* protokol koji služi za distribuciju *link-state* topologije vanjskim entitetima u mreži. *Path Computation Element protocol (PCEP)* koristi se u *Multiprotocol Label Switching (MPLS)* IP mrežama za centralnu kontrolu puta, no najčešće korišteni protokol *southbound* sučelja je Openflow. Temeljna uloga Openflow protokola nije direktna konfiguracija mrežnih uređaja, već implementacija pravila na podatkovnu ravan infrastrukturnog sloja mreže. Implementirana pravila određuju „sudbinu" svakog paketa na svom putu kroz mrežu.[21]

Osnovne komponente koje definira Openflow specifikacija su Openfow preklopnik i Openflow kontroler. Kroz te dvije komponente odvojene su podatkovna i kontrolna ravan.[22]

Openflow preklopnik je u biti jednostavan fizički ili virtualni uređaj koji se može pokretati na uobičajenom hardveru, a sadrži jednu ili više tablica unutar koje se pohranjuju zapisi koje je definirao kontroler. Ti podaci govore preklopniku što treba učiniti s paketima koji odgovaraju određenoj klasifikaciji.

Opeflow kontroler je softverska komponenta koja se pokreće na jednom ili više odvojenih poslužitelja koji su s preklopnicima spojeni na zasebnom mrežnom segmentu putem kojeg se odvija Openflow komunikacija. U skladu i s imenom, glavna zadaća kontrolera je obavljati sve zahtjevne komputacijske zadaće L2 i L3 protokola te u skladu s rezultatima ažurirati prethodno spomenute tablice na preklopnicima.

Glavna prednost koju ovakva separacija slojeva u SDN mreži nudi je mogućnost implementacije raznih vrsta aplikacija te razvoj aplikacija koje su neovisne o hardveru koji ih pokreće, ali prije svega korištenje jeftinog i lako zamjenjivog hardvera na podatkovnom sloju mreže.

3.3. Uloga i karakteristike kontrolera u radu SDN-a

SDN kontroler je softverski sustav koji omogućava upravljanje stanjem u mreži sa centralne lokacije. Kontroler može funkcionirati kao zasebna jedinica ili kao visoko dostupan uređaj

[21] Complete practical SDN and Openflow fundamentals
[22] Software defined Networks, poglavlje 2.

gdje je jedan kontroler *master*, a ostali su *slave*. U slučaju prestanka rada *master* kontrolera jedan od *slave* kontrolera može preuzeti funkciju primarnog kontrolera.

Temeljna zadaća kontrolera u radu SDN-a očituje se kroz sljedeće funkcionalnosti:

- Upravljanje i distribucija stanjem u mreži – da bi kontroler mogao adekvatno upravljati s ostalim uređajima u mreži, mora imati integriranu bazu podataka u koju će zapisivati podatke o ostvarenim konekcijama, stanju u mrežnoj topologiji te internoj konfiguraciji
- Omogućavanje pristupa mrežnim resursima putem modernog *norhbound* API-ja; taj API je u većini slučajeva baziran na RESTful arhitekturi, putem REST sučelja kontroler pruža aplikacijama pristup resursima u mreži tako da aplikacijama sakrije svu kompleksnost nižih slojeva te da prevodi naredbe više razine u Openflow pravila putem kojih se ostvaruje komunikacija sa SDN preklopnicima
- Ostvarivanje veze s agentima na SDN preklopnicima, usmjernicima ili *load balancerima*
- Podržavanje jednog ili više protokola *southbound* sučelja – danas se primarno za to koristi Openflow protokol
- Mehanizam otkrivanja mrežne topologije i upravljanje putanjama u mreži.[23]

U nastavku će ukratko biti opisan rad nekoliko najčešće korištenih kontrolera u industriji.

3.3.1. Aruba HP VAN SDN kontroler

Aruba HP VAN SDN kontroler je vlasnički (engl. *proprietary*) Openflow kontroler američke kompanije Hewlett-Packard. Softver je prije svega zamišljen kao suplement HP Aruba preklopnicima i usmjernicima koji osim tradicionalnih L2 i L3 protokola nude podršku i za Openflow. Osim podrške za vlastite preklopnike, ovaj kontroler nudi podršku za sve preklopnike koji podržavaju Openflow protokol. Aruba VAN SDN kontroler dizajniran je da radi u raznim okolinama, što uključuje kampus mreže, podatkovne centre, mreže pružatelja telekomunikacijskih usluga te unutar privatnih i javnih *cloud* okruženja. Radi se o vrlo stabilnom proizvodu s iscrpnom i kvalitetnom dokumentacijom i vizualnim sučeljem s velikim brojem opcija, uključujući i preglednik toka podataka. Upravo zbog te

[23] SDN: Software Defined Networks, str. 73

činjenice Aruba HP VAN SDN kontroler bit će korišten u praktičnom dijelu ovog rada u kojem će biti važno vizualno pratiti tijek naših podataka kroz mrežu.

HP nudi podršku za razvoj aplikacija od strane trećih strana kroz RESTful arhitekturu, a osim toga, razvio je i tri svoje aplikacije koje se već sada koriste u mnogim okolinama. Te tri aplikacije su:

- **HPE Network Optimizer** – ova aplikacija razvijana je u suradnji s inženjerima iz Microsofta. Koristi se kod uspostave i održavanja Skype poziva. Prilikom inicijalizacije poziva Skype aplikaciji šalje detalje o sesiji, što uključuje izvornu i odredišnu adresu te potrebe za propusnošću mreže na temelju kojih Network Optimizer prilagođava razinu kvalitete usluge u mreži koristeći Openflow protokol[24]
- **HP Network Protector** – aplikacija koja se može koristiti za centralnu distribuciju sigurnosnih politika u realnom vremenu koristeći Openflow zapise[25]
- **HP Distributed Cloud Networking** – omogućuje pružateljima usluga i tvrtkama izgradnju distribuiranog okruženja s više oblaka na jednostavan, otvoren i agilan način pomoću SDN-a i virtualizacije mreže.[26]

Budući da se radi o vlasničkom sustavu tvrtke HP, ovaj kontroler može se koristiti besplatno samo kao demo verzija.

3.3.2. Ryu kontroler

Ryu je okvir koji nudi cjelokupan set biblioteka i alata potrebnih za razvoj SDN aplikacija. Ryu okvir razvio je japanski telekomunikacijski div NTT na programskom jeziku Python, a temeljna komponenta okvira je Ryu kontroler.

Ryu dolazi kao softver otvorenog koda pod Apache 2.0 licencijom.

Kao i kod svih ostalih kontrolera, Ryu je na *southbound* sučelju komunicira s infrastrukturnim mrežnim uređajima i tu osim Openflowa (1.0 – 1.5) podržava također i NETCONF te OF-CONFIG protokole. Kao i ostali kontroleri na *northbound* sučelju, Ryu nudi aplikacijama pristup uređajima putem RESTful arhitekture.

---

[24] https://partnersolutions.skypeforbusiness.com/solutionscatalog/networking-hardware-infrastructure/hpe-network-optimizer, 30.07.2018

[25] http://h17007.www1.hpe.com/si/en/networking/solutions/technology/sdn/portfolio.aspx#.W1974tgzbOQ, 30.07.2018

[26] Ibidem

Da bi Ryu radio, potrebno je instalirati Python pakete na nekom od Linux distribucija. Uz kontroler dolazi čitav set aplikacija koje je moguće pokrenuti. Neke od njih su *L2 Learning Switch,* aplikacija koja simulira tradicionalno prosljeđivanje datagrama na L2 razini, *Router,* koji simulira distribuirani L3 uređaj, i *Firewall* koji simulira rad L4 uređaja. Aplikacije se mogu pokretati u isto vrijeme, a komunikaciju između ostvaraju putem asinkronih poruka.[27]

Ryu kontroler nudi osnovno grafičko sučelje na kojem je moguće vizualizirati topologiju mreže, ali uz minimalnu količinu informacija. Većinu konfiguracije potrebno je vršiti modificiranjem tekstualnih datoteka te Ryu zbog toga često ne koriste početnici ili developeri i administratori koji nisu upoznati s jezikom Python.

Ryu kontroler koristi Američka agencija za Nacionalnu sigurnost (engl. NSA) za potrebe kontrole prometa u svojoj internoj mreži. Aplikacija koju koriste za to napisana je u samo nekoliko tisuća linija koda u Pythonu.[28]

3.3.3. Opendaylight kontroler

Opendaylight kontroler dio je Opendaylight projekta koji su razvijale vodeće tvrtke u IT industriji pod okriljem *Linux Foundation* organizacije. Više od 40 organizacija uključeno je u njegov razvoj, uključujući Brocade, Cisco, HP i IBM, a sve radi doprinosa cjelokupnom razvoju inovacija u polju SDN-a.

Opendaylight nudi otvorenu platformu za razvoj SDN-a te ima veliku podršku i industrije i developera. Ono što Linux znači za operacijske sustave, to Opendaylight znači za računalne mreže, otvorenost i ubrzani razvoj kroz kolaboraciju. Riječ je o trenutačno vodećoj platformi za razvoj softverski definiranih mreža.[29]

Opendaylight je pisan u Javi, što znači da se može pokrenuti na velikom broju operacijskih sustava, ali tipično dolazi kao zaseban Linux *appliance* koji se može importirati kao virtualna mašina unutar *hypervisora*.

Opendaylight također nudi podršku velikom broju protokola *southbound* sučelja. Protokoli se kao dodaci dinamički vežu s apstrakcijskim slojem (SAL). Apstrakcijski sloj služi kao

---

[27] https://thenewstack.io/sdn-series-part-iv-ryu-a-rich-featured-open-source-sdn-controller-supported-by-ntt-labs, 31.07.2018

[28] https://www.networkworld.com/article/2937787/sdn/nsa-uses-openflow-for-tracking-its-network.html, 31.07.2018

[29] https://thenewstack.io/sdn-series-part-vi-opendaylight, 31.07.2018

ljepilo koje veže module višeg reda s modulima nižeg reda koji definiraju sam *southbound* protokol koji će se koristiti. To, naravno, ovisi o mogućnostima uređaja infrastrukturnog sloja te o konfiguraciji. Unutar kontrolera nalazimo module za različite svrhe kao što su modul za otkrivanje topologije, modul za kolekciju statistike i modul za programiranje uređaja, a svaki od njih odgovara na pozive koje dobiva od apstrakcijskog sloja.

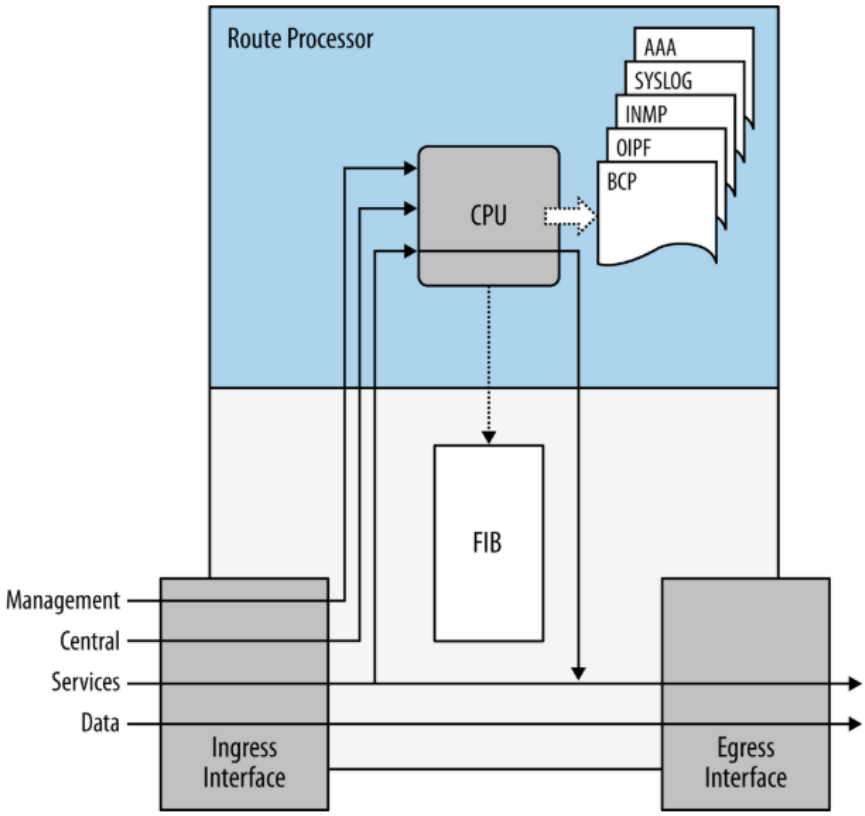
Na *northbound* sučelju Opendaylight daje podršku mrežnim aplikacijama koristeći RESTful arhitekturu za aplikacije koje se nalaze na istom ili odvojenom mrežnom segmentu te na *OSGi* okviru za interne JAVA aplikacije koje se nalaze implementirane na istom sklopovlju na kojem se nalazi i kontroler.

3.4. Upravljačka i podatkovna ravan

Sama bit SDN-a leži u mehanizmu razdvajanja kontrolne i podatkovne ravni. Usmjernici i preklopnici u tradicionalnoj mreži imaju kontrolnu i podatkovnu ravan implementirane unutar sebe, ali na zasebnim dediciranim procesorima. SDN naprave odvajaju kontrolnu i podatkovnu ravan na zasebne uređaje te komunikaciju između njih ostvaruju upotrebom zasebnog SDN protokola.

Slika 5 prikazuje tipičnu arhitekturu preklopnika/usmjernika tradicionalne mreže. Donji dio slike prikazuje podatkovnu ravan koja je implementirana na zasebnoj kartici s dediciranim procesorima za obradu podataka fizičkih sučelja. Podatkovna ravan sadrži tablicu u kojoj su definirana pravila za usmjeravanje paketa (engl. FIB), a tablicu popunjava kontrolna ravan ovisno o kalkulacijama protokola više razine, na slici je vidimo kao CPU. Kada na portove stignu paketi koji nemaju definirana pravila, oni se automatski prosljeđuju procesoru na daljnju obradu. Podatkovna ravan povezana je s centralnim procesorom preko brze interne sabirnice. Radi osiguravanja veće dostupnosti, kontrolna ravan često se implementira kao redundantna jedinica.[30]

[30] SDN: Software Defined Networks, poglavlje 2.



Slika 3.4 Kontrolna i podatkovna ravan u tradicionalnoj mreži[31]

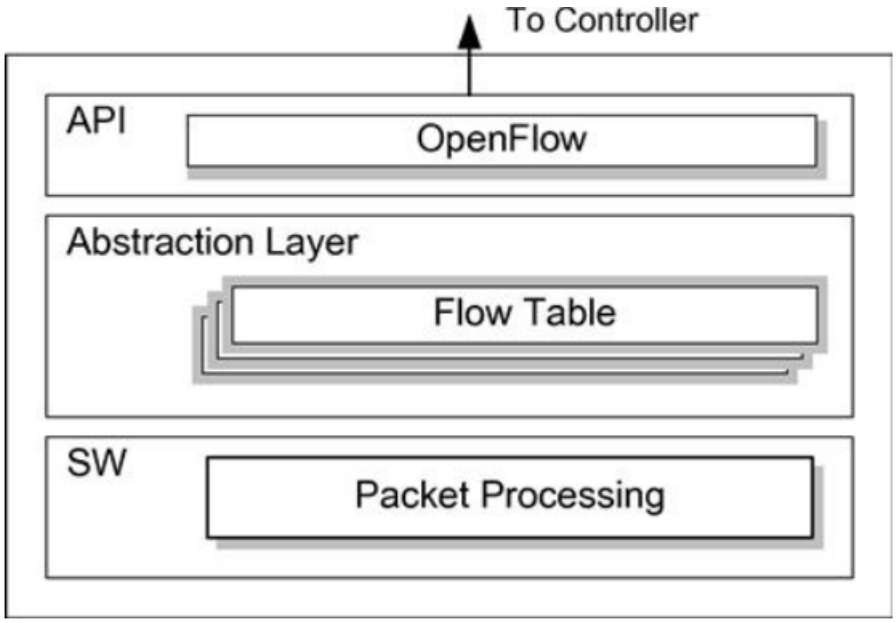
Sklopovi i programi podatkovne i kontrolne ravni u tradicionalnim mrežama distribuirane su naravi. Zbog brze komunikacije koju podatkovna i kontrolna ravan ostvaruju preko internih sabirnica te efikasnih ASIC čipova razvijenih posebno za obavljanje određenih funkcija, ovakav oblik umrežavanja ima svojih prednosti u vidu brzine, pouzdanosti i učinkovitosti, ali i mane kao što su distribuirana kontrola te zatvoreni razvoj čipova, što povećava cijenu uređaja te usporava inovacije i razvoj jer je sama funkcionalnost integrirana direktno u hardver čiji je razvoj u potpunosti pod kontrolom proizvođača uređaja.

Kod SDN mreže odgovornost kontrolne i podatkovne ravni odvojena je na zasebnim uređajima. Preklopnici i usmjernici su pojednostavnjeni te oni više ne moraju sadržavati čitav niz kompleksnih kontrolnih jedinica i kodova, a te funkcije nalaze se implementirane u SDN kontroleru s kojim komunikaciju ostvaruju na zasebnom upravljačkom sučelju. Kontroler je zadužen za sve kompleksne kalkulacije i programiranja te davanje instrukcija preklopnicima u jednostavnom obliku. Kontroler sada može upravljati većim brojem

[31] T.D. Nadeau, K. Grey, SDN: Software Defined Networks, poglavlje 2.

preklopnika s jednog mjesta, a upravo za tu vrstu komunikacije najčešće se koristi Openflow protokol.

Openflow protokol definira komunikaciju između kontrolera i preklopnika kao medijator između kontrolne i podatkovne ravni u SDN mreži, što omogućuje kontroleru da programira preklopnik unosom *flow* zapisa u tablice koje se nalaze na SDN uređajima. Svaki preklopnik ima implementiranu jednu ili više *flow* tablica, ovisno o verziji Openflow protokola koja se koristi. Osim tablice, u preklopnik je ugrađen i agent za komunikaciju prema kontroleru te softver za procesuiranje paketa.



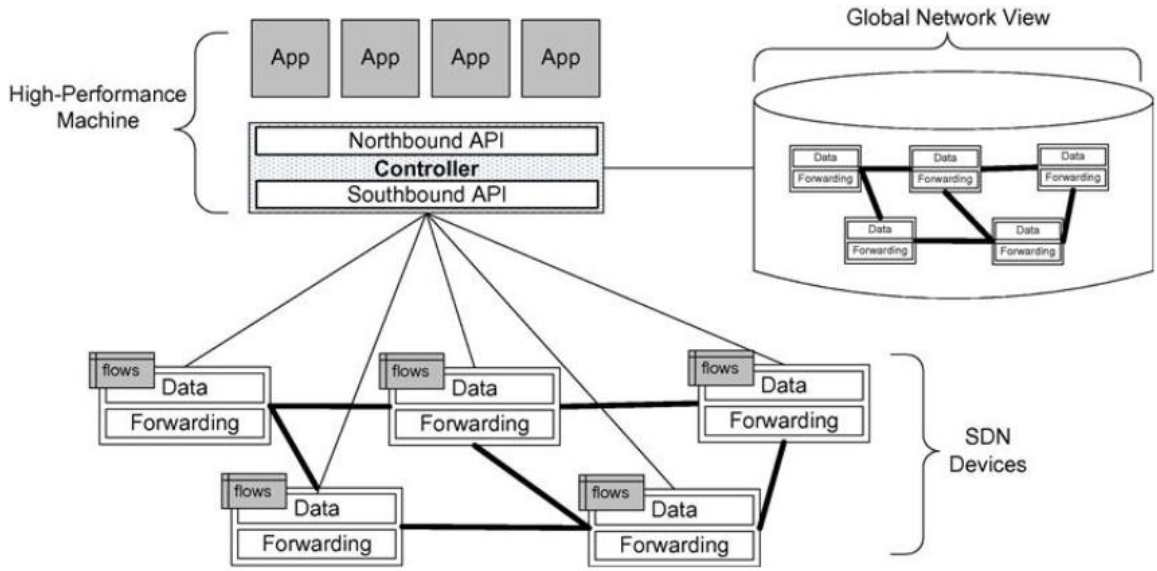
Slika 3.5 Anatomija SDN preklopnika[32]

Svaku tablica čini niz *flow* zapisa koji se sastoje od tri vrste polja, zaglavlja, brojača i akcije. Kada preklopnik primi određeni paket, pregledat će zapise u svojim *flow* tablicama. Ako pronađe određeni zapis koji odgovara nekom od zaglavlja paketa, izvršit će akciju koja je definirana u zapisu i ažurirati brojač, u suprotnom paket će biti odbačen. Pritom je bitan prioritet zapisa, zapis s većim prioritetom uvijek ima prednost. Preklopnik paket može odbaciti, proslijediti ga na određeno sučelje, tj. na grupu sučelja ili ga proslijediti kontroleru na daljnju analizu i obradu. Svaki zapis ima definirano vrijeme trajanja koje se unutar Openflowa određuje na razini *Idle timeout* vrijednosti te *Hard timeout* vrijednosti. *Idle timeout* određuje u kojem će vremenskom razdoblju *flow* zapis biti izbrisan iz *flow* tablice preklopnika ako se ne dogodi nijedan *match* koji odgovara tom zapisu. *Hard timeout* definira

[32] C. Black, P. Goransson, Software Defined Networks, poglavlje 4.

koliko će neki zapis dugo biti zapisan u *flow* tablici preklopnika bez obzira na to je li se nad njim dogodio *match* ili nije.

Prema Openflow v1,0 specifikaciji, jednom kada je zapis zapisan u tablicu, preklopnik može napraviti *match* nad sljedećim poljima u *ethernet zaglavlju:* VLAN ID, VLAN prioritet, izvorna MAC adresa, odredišna MAC adresa, tip *payloada,* dolazna IP adresa, odlazna IP adresa, IP protokol, ToS, TCP/UDP izvorni port te TCP/UDP odredišni port. Naknadne verzije protokola definiraju i druga polja koja je moguće *matchirati* kao što je MPLS oznaka, DSCP vrijednost i drugi.



Slika 3.6 Prikaz separacije ravni u SDN mreži[33]

Slika 3.6 Prikaz separacije ravni u SDN mrežiPrikazuje način separacije ravni u SDN mreži. Komunikacija između kontrolera kao uređaja kontrolne ravni i preklopnika kao uređaja podatkovne ravni teče uz pomoć modifikacije *flow* pravila na *southbound* sučelju SDN mreže.

3.5. Implementacija i konfiguracija

Implementacija i konfiguracija uređaja tradicionalne mreže vrši se pristupom tekstualnom sučelju uređaja (engl. CLI) ili grafičkom korisničkom sučelju uređaja (engl. GUI) te unosom ručnih naredbi u slučaju pristupa tekstualnom sučelju ili postavljanjem parametara unutar

[33] C. Black, P. Goransson, Software Defined Network, poglavlje 4.

grafičkog sučelja. Administratori najčešće uređajima pristupaju putem SSH ili TELNET protokola, a naredbe unose u tekstualnom obliku. Konfiguracije se primjenjuju odmah kod unosa jer se promjene unose direktno u radnu memoriju uređaja, kako bi konfiguracija ostala permanentna, i nakon ponovnog pokretanja potrebno ju je upisati u trajnu memoriju uređaja. S obzirom na distribuiranu narav kontrolne ravni u tradicionalnim mrežama, svaki je uređaj potrebno konfigurirati zasebno. Takav proces traži puno vremena administratora jer ne postoji mogućnost konfiguracije s centralne lokacije. Kod većih promjena u konfiguraciji potrebno je spajati se na desetke uređaja te na svakom od njih vršiti izmjene. Zbog samog obujma posla u velikim organizacijama takve promjene radi veći broj administratora, a između administratora treba postojati adekvatna komunikacija. No bez obzira na to, zbog ručnog karaktera konfiguracije, postoji velika mogućnost pogreške, što može izazvati velike probleme u radu mreže.

Različiti proizvođači mrežnih uređaja definiraju i različite naredbe putem kojih se vrši konfiguracija uređaja. Razlog je tomu nepostojanje standardnog operacijskog sustava. Svaki proizvođač implementira vlastiti sustav u uređaje koje proizvodi, a to zahtijeva polivalentnost i široko znanje administratora koji administriraju cjelokupnu mrežu.

SDN rješenja imaju odvojene kontrolnu i podatkovnu ravan, što znači da jedan mrežni operacijski sustav ili više njih, ako se radi o *clusteru,* upravljaju većim brojem mrežnih uređaja. Kod implementacije potrebno je odrediti gdje ćemo smjestiti kontroler. U teoriji on se može nalaziti i izvan lokacije na kojoj nam se nalaze krajnji uređaji, ali preporučljivo ih je postaviti na istu lokaciju, što bliže uređajima koji će prosljeđivati pakete kako bi se smanjilo vrijeme odziva mreže.

Kontroler se u najčešćem broju slučajeva implementira kao virtualna mašina na *hipervizoru* i spajamo ga s ostalim uređajima na posebnom upravljačkom TCP segmentu.

Kod konfiguracije mrežnih uređaja potrebno je definirati portove koji će služiti za upravljačku komunikaciju, staviti ih u zaseban segment (engl. *bridge*) i onda ih povezati s kontolerom. Na isti način potrebno je definirati i *bridge* u koji ćemo spajati krajnje uređaje. Nakon što je uređaj povezan s kontrolerom, svu ostalu konfiguraciju možemo raditi sa strane aplikacija putem *northbound* sučelja.

Ovakav pristup implementaciji i konfiguraciji znači da je na mrežnim uređajima potrebno izvršiti minimalan set naredbi prije nego što ga spojimo s kontrolerom i pustimo u produkciju.

Same aplikacije zadužene su za virtualizaciju mrežnih funkcija. Isti uređaj možemo koristiti kao vatrozid i preklopnik u isto vrijeme, a to je nešto što u tradicionalnoj mreži nije moguće implementirati.

Danas se prilikom implementacije SDN mreže postupak migracije gotovo uvijek radi postupno, a većina softverskih i hardverskih preklopnika nudi hibridnu opciju rada. Kod hibridne opcije rada preklopnik se u isto vrijeme može ponašati kao uređaj tradicionalne mreže i kao uređaj SDN mreže. U Openflow okruženju na dnu *flow* tablice implementiran je *wildcard* zapis s najnižim prioritetom. Ako se ne dogodi *match,* preklopnik će paket proslijediti prema *NORMAL cjevovodu*, tj. na tradicionalno procesuiranje. Možemo reći da je SDN funkcionalnost u ovom slučaju implementirana kao zaseban sloj iznad tradicionalne mreže (engl. *Overlay).*

3.6. Upravljanje i održavanje

U svrhu upravljanja i održavanja tradicionalnom mrežnom infrastrukturom koriste se specijalizirani alati kao su PTRG, Nagios, Zabbix te Solarwind Network Manager. Na tržištu postoji još niz drugih rješenja, a svake godine izlaze i novi, no navedeni alati su među najčešće korištenima. Mrežama srednje velikih i velikih organizacija gotovo je nezamislivo upravljati bez nekoga od njih, a ti alati prvenstveno su dizajnirani tako da prate propusnost, dostupnost i performanse mreže i mrežnih komponenata.[34]

Administratorima koji prate mrežu prije svega je bitno da bilo u kojem trenutku mogu dobiti uvid o svim resursima koji se nalaze u mreži, a još je bolje ako su te informacije prikazane topografski u vizualnom obliku. Alati za upravljanje mrežom otkrivaju veze između mrežnih uređaja korištenjem SNMP, CDP, LLDP i ICMP protokola.

Druga važna karakteristika ovih alata jest mogućnost prilagodbe kontrolne ploče ovisno o potrebama IT odjela unutar organizacije. Također, s obzirom na to da mnoge organizacije koriste mrežnu opremu od različitih proizvođača, alati za upravljanje, kontrolu i održavanje mreže moraju davati podršku za te proizvode, a takva podrška često dolazi već integrirana u sustav ili u vidu predložaka koji se mogu skinuti s Interneta i odmah implementirati.

[34] https://www.paessler.com/solutions, 04.08.2018

Obavještavanje o događajima u mreži vrlo je važna funkcija koju ovi alati pružaju. Razumljivo je da mrežni administratori ne mogu čitavo svoje radno vrijeme provesti promatrajući aktivnosti koje se događaju na kontrolnoj ploči, zato alati za upravljanje, kontrolu i održavanje mreže pružaju mogućnost konfiguracije obavještavanja o bitnim promjenama u mreži. Administratori tako mogu pravodobno reagirati i otkloniti problem koji je primijećen. U većini slučajeva obavijesti se šalju na e-mail, ali je moguće dodati i druge opcije, kao što je Short message service (SMS).

Automatsko upravljanje uređajima u mreži moguće je korištenjem alata kao što su Solarwinds Configuration Manager ili Ansible. Nakon konfiguracije ti se alati mogu spajati na CLI sučelja uređaja koristeći SSH protokol i vršiti izmjene u konfiguraciji, a posebno su učinkoviti kada je potrebno izvršiti automatizaciju konfiguracije više uređaja u što kraćem roku. Naravno, takve je aktivnosti moguće zakazati za određeno vrijeme, npr. kada je mreža najmanje opterećena.[35]

Budući da je SDN relativno nova tehnologija koja mijenja način na koji će mreže funkcionirati u budućnosti, navedeni alati za upravljanje i održavanje mrežom NMS (engl. Network Management Systems) još uvijek ne daju potpunu podršku za SDN. Promjene koje će se morati dogoditi jesu da će NMS alati morati uspostaviti nova sučelja koja će moći komunicirati s mrežnim kontrolerima.

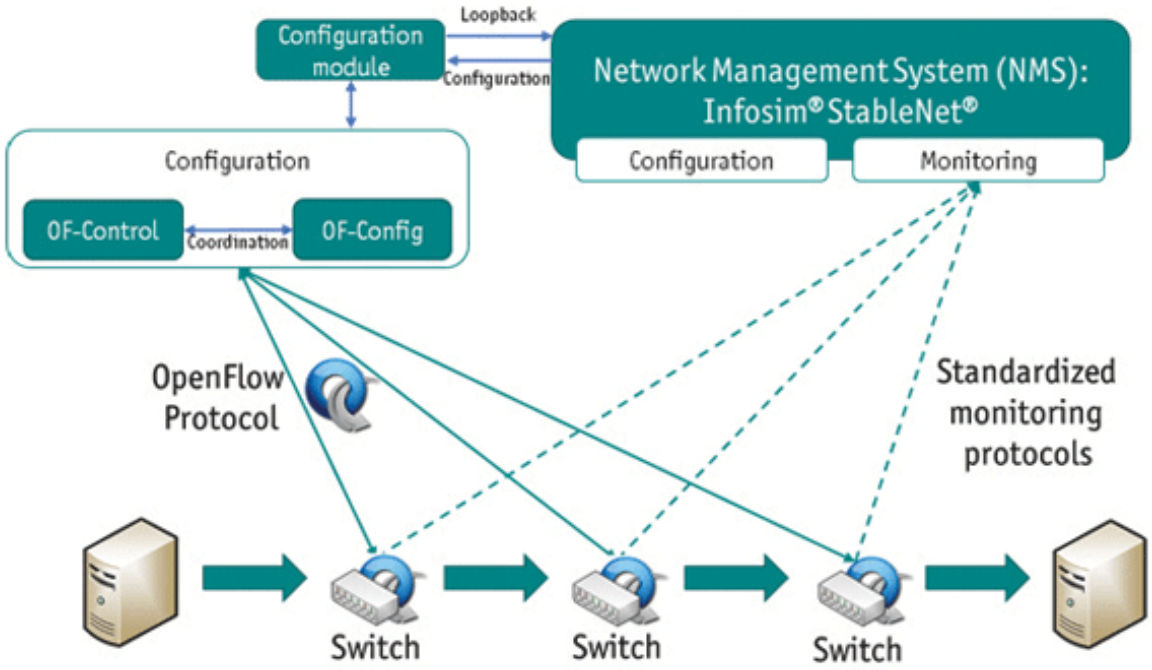
Budući da je kontroler centralno mjesto SDN mreže, NMS alati će ostvarivanjem komunikacije s kontrolerom dobivati sljedeće važne informacije o mreži:

- automatsko otkrivanje mrežne topologije te promjena unutar nje
- upravljanje konfiguracijom mrežnih uređaja
- prikupljanje događaja o uređajima u mreži (greške, performanse).

Rješenje za upravljanje i održavanje SDN mrežom StableNet of tvrtke Infosim uz podrške za uređajima tradicionalne mreže nudi i podršku za SDN uređaje upravo na opisani način. StableNet ostvaruje dvosmjernu komunikaciju sa SDN kontrolerom, a tako omogućava gore opisane funkcije. Administratorima se omogućava da s jednog centralnog mjesta vrše konfiguraciju mreže definiranjem *flow* pravila koje SDN kontroler prosljeđuje preklopnicima, a s druge strane u realnom vremenu administratorima prikazuju podatke

[35] https://www.solarwinds.com/network-configuration-manager, 04.08.2018

prikupljene iz brojača u Openflow protokolu. Prikaz opisanog rada StableNet NMS sustava prikazan je na Slika 3.7.



Slika 3.7 Prikaz rada StableNet sustava[36]

3.7. Trošak za organizaciju

Kod definiranja troškova koje organizacije čine kod ulaganja u IT bitne su nam dvije temeljne kategorije istih:

- kapitalni troškovi (CAPEX)
- operativni troškovi (OPEX).

Kapitalni troškovi su troškovi koje organizacija čini prilikom nabave opreme koja će se koristiti dulje vrijeme.

Operativni troškovi su troškovi koje organizacija čini kod svakodnevnog rada i upravljanja.[37]

[36] https://telnetnetworks.wordpress.com/2014/10/30/sdn-and-network-management-infosim-stablenet-current-state-of-affairs-and-roadmap/, 04.08.2018
[37]https://www.investopedia.com/ask/answers/020915/what-difference-between-capex-and-opex.asp, 05.08.2018

Razumno je pretpostaviti da će za svaku organizaciju glavna odrednica o tome kada i kako migrirati s tradicionalne mreže na SDN mrežu troškovi biti ključan i odlučujući faktor, što je razumljivo jer ipak živimo u tržišnoj ekonomiji unutar koje su smanjenje troškova i povećanje profita temeljni pokretači većine poslovnih odluka i procesa.

Gledano s aspekta troškova IT-a, važna je činjenica da su kapitalni troškovi koje srednje velike i velike organizacije čine kupujući poslužiteljsku procesorsku snagu puno manji naspram kupnje iste procesorske snage mrežne opreme. Razlog je tomu prije svega vlasnička narav mrežnih uređaja, komponenti i softvera.[38]

Jasno je da taj *status quo* u budućnosti ne može opstati, a još je jasnije zašto u upravljačkom vijeću ONF-a sjedi tako velik broj predstavnika kompanija kao što su Facebook, Microsoft i Goldman Sachs[39]. Upravo zbog visokih kapitalnih i operativnih troškova na mrežnu opremu i održavanje mreže oni moraju podržavati razvoj protokola i sklopovlja koji će njihove kapitalne i operativne troškove smanjiti na najmanju moguću razinu.

Načini na koji će migracija s tradicionalne mreže na SDN smanjiti kapitalne troškove jesu smanjena ovisnost o opremi određenog proizvođača, smanjena cijena softvera zbog centralizirane naravi kontrolne ravni u SDN mrežama i veća iskoristivost linkova korištenjem aplikacija za upravljanje prometom.

Načini na koji se mogu smanjiti operativni troškovi su smanjena potreba za ljudskim resursima zbog centralizirane naravi upravljanja i konfiguracije, bolja iskoristivost energije preko cijele mreže s obzirom na to da preklopnici i usmjernici više ne moraju napajati lokalne instancije kontrolne ravni, smanjeni troškovi popravka, zamjene, testiranja i stavljanja u pogon postojećih i novih preklopnika.

3.8. Koristi za organizaciju

Trenutačno najveća prednost korištenja komponenti tradicionalne mreže jest u činjenici da se s kupnjom nekog od uređaja poznatih proizvođača osim marke kupuje i podrška. Velikim korporacijama kao što je Google ta činjenica nije bitna kao manjim organizacijama koje nemaju organizacijsku sposobnost pružanja interne podrške i razvoja.[40] Uza sve to

[38] T.D. Nadeau, K. Gray, SDN, Software Defined Networks, poglavlje 2., str. 24
[39] Chuck Black, Paul Goransson, Software Defined Networks, poglavlje 9.
[40] Ibidem

tehnologije tradicionalne mreže, iako spore u razvoju, izdržale su test vremena i pokazale se stabilnim i pouzdanim proizvodom tijekom posljednjih nekoliko desetljeća.

Najveće prednosti koje SDN u budućnosti može ponuditi organizacijama jesu:

- **Centralizirano upravljanje** – s obzirom na to da SDN odvaja kontrolnu i podatkovnu ravan, omogućava povećanu iskoristivost postojećih linkova tako ostvaruje veću efikasnost mreže za istu ili manju cijenu. U isto vrijeme, uz korištenje adekvatnih alata za virtualizaciju mreže, omogućuje se centralizirano upravljanje kompletnom fizičkom i virtualnom mrežnom infrastrukturom
- **Povećana razina sigurnosti** – s obzirom na centralizirano upravljanje i nadzor, lakše je sakupljati podatke o mrežnom prometu na jednom mjestu te reagirati prema potrebi
- **Smanjena kapitalna i operativna ulaganja** – kao što je definirano u prethodnom naslovu, SDN dugoročno smanjuje operativne i kapitalne troškove
- **Infrastruktura spremna za računarstvo u oblaku** – iskorištavanjem mogućnosti koje nude SDN tehnologije povećava se skalabilnost mreže, što znači smanjeno vrijeme potrebno za proširenje trenutnih operacija[41]
- **Proširivost** – s obzirom na otvorenu narav SDN tehnologija, svaka organizacija može razvijati vlastite aplikacije s obzirom na interne potrebe
- **Kvalitetnije planiranje** – s obzirom na kvalitetniji uvid u topologiju mreže, olakšava se uvid u trenutačni portfelj svih mrežnih resursa te planiranje za promjenama u budućnosti
- **Omogućavanje eksperimentiranja na produkcijskoj mreži** – korištenjem hibridnih preklopnika moguće je implementirati SDN mrežu na uređajima tradicionalne mreže i simultanim pokretanjem jedne i druge opcije isprobavati mogućnosti SDN mreže bez ugrožavanja trenutačnog rada mrežne infrastrukture.

[41] https://www.cisco.com/c/en/us/solutions/software-defined-networking/benefits.html, 06.08.2018

4. Praktični dio rada – metodologija, kriteriji, usporedbe i vrednovanja

L2 mreže najčešće su korišteni oblik umrežavanja u malim i srednje velikim organizacija, a u takvim uvjetima ne koristi se velik broj usmjernika već se najveći dio komunikacije ostvaruje korištenjem L2 okvira. Prednosti korištenja ovakvih mreža jesu manja cijena nabave i održavanja uređaja, veća brzina prosljeđivanja paketa, a samim time i manja latencija u mreži.

Primarni kriterij usporedbe između SDN i tradicionalne mreže bit će usporedba mogućnosti upravljanja prometom unutar L2 *broadcast* domene. U slučaju tradicionalne mreže upravljanje prometom vršit će se manipulacijom STP i RSTP topologije promjenom STP prioriteta preklopnika, što će prisiliti konvergenciju mreže i promjenu putanje paketa. Za potrebe manipulacije putanjom u SDN mreži bit će prikazan rad SDN aplikacije u vidu interaktivne BASH skripte koju sam napisao kako bih na lakši način, putem *northbound* sučelja uz pomoć *RESTful* poziva, vršio implementaciju *flow* zapisa u *flow* tablice preklopnika i tako s centralne lokacije određivao putanju prometa paketa ovisno o odabranim parametrima.

Mjerenje brzine konvergencije izvršit će se pokretanjem *ping* sonde od izvornog uređaja prema odredišnom uređaju unutar unaprijed definiranog razdoblja.

Konvergencija mreže u prvoj topologiji bazirana je na STP protokolu, i to u dva mjerenja – jedno za STP, a drugo za RSTP protokol. Konvergencija mreže u SDN topologiji ovisi o programiranju tokova podataka na centralnom poslužitelju.

Mjerenje će se izvršiti na sljedeći način: mjerit će se broj paketa za koje izvorna strana nije primila adekvatni ICMP odgovor te će se kao dodatni kriterij mjeriti vrijeme u sekundama unutar kojeg nije postojala komunikacija između izvornog i odredišnog uređaja. U slučaju da za svaki poslani paket primimo i adekvatan odgovor te da vrijeme nedostupnosti mreže nije mjerljivo ručnim pokretanjem brojača, kao vrijeme nedostupnosti mreže uzet će se maksimalna latencija paketa u mreži. Prilikom ispisa rezultata bit će prikazan broj paketa za koje nije primljen odgovarajući odgovor, vrijeme nedostupnosti mreže te minimalna, prosječna i maksimalna latencija u mreži.

Očekivano je da će konvergencija u tradicionalnoj mreži korištenjem STP protokola trajati između 30 i 50 sekundi, konvergencija korištenjem RSTP protokola trebala bi trajati manje od 6 sekundi, dok bi konvergencija mreže u SDN okolini trebala trajati manje od 2 sekunde, minimalna, prosječna i maksimalna latencije u mreži trebale bi biti podjednake i kretati se u rasponu od 2 pa do maksimalno 60 milisekundi.

Vrednovanje rezultata izvest će se prema načelu 'manje je bolje'. Manji broj neuspješno zaprimljenih ICMP *echo reply* paketa te kraće razdoblje bez komunikacije između uređaja smatrat će se povoljnijim. Isto vrijedi i za minimalnu, prosječnu te maksimalnu latenciju u mreži.

U istom dijelu praktičnog rada bit će prikazane usporedbe načina vizualizacije mrežne topologije kod tradicionalne i SDN mreže te usporedba vremena automatskog čitanja podataka i generiranja dijagrama o topologiji u mreži. Usporedba će se izvršiti pokretanjem odbrojavanja potrebnog za dobivanje trenutnog dijagrama mreže. Manji rezultat smatrat će se povoljnijim.

4.1. Opis testnih topologija

Za potrebe rada unutar GNS3 mrežnog simulatora bit će implementirane dvije odvojene mrežne topologije, dodatno kao *hipervizor* za potrebe virtualizacije koristit će se *Vmware Workstation 12 Pro*. Prva topologija simulirat će rad tradicionalne mreže, dok će druga topologija simulirati rad SDN mreže. Obje mreže bit će logički i fizički identično implementirane, i to bazirano na troslojnoj mrežnoj hijerarhiji s jednim jezgrenim, dva agregatna i tri pristupna preklopnika. U svaki od pristupnih preklopnika bit će spojene po jedna virtualna mašina, dvije CENTos7 virtualne mašine te jedna Windows Server 2016 virtualna mašina, a te će mašine simulirati rad krajnjih uređaja te će se koristiti za testiranja. Dodatno će u preklopnik jezgre biti spojen jedan virtualni usmjernik s instaliranom Cisco IOS preslikom. Za testiranje brzine konvergencije koristit će se virtualna mašina CentOS1 i usmjernik R1, a virtualna mašina WinServer2016 koristit će se za dobivanje informacije o topologiji tradicionalne mreže. Slijedi detaljniji opis svake od topologija, uključujući korištene konfiguracije te grafički prikaz mreže iz GNS3 mrežnog simulatora.

4.1.1. Tradicionalna mreža

U topologiji tradicionalne mreže implementirano je šest virtualnih preklopnika unutar GNS3 mrežnog simulatora (GNS3 Verzija 2.1.5). Virtualni preklopnici pokreću Cisco IOSvL2 15.2(20170321:233949) operacijski sustav i spojeni su u hijerarhijsku mrežnu topologiju kao što je prikazano na Slika 4.1. Na svakom od preklopnika podešena je osnovna konfiguracija, uz ručno definiranje da inicijalno želimo koristiti standardni STP protokol (Cisco prema zadanom ime uključen *RSTP* protokol), a iza svake konfigurirane naredbe označeno je komentarom što ona predstavlja:

```
enable !ulazak u user mod
configure terminal !ulazak u exec mode
hostname IMEPREKLOPNIKA !definiranje imena preklopnika
no ip domain-lookup !isključivanje DNS lookup-a
line console 0 !ulazak u konfiguraciju konzolne linije
      no logging syncrounous !isključivanje sinkronizacije
ispisa
spanning-tree mode pvst !uključivanje standardnog STP-a
```

Kôd 4.1 Osnovna konfiguracija preklopnika u topologiji tradicionalne mreže

Linkovi između preklopnika podešeni su kao *trunk linkovi*, dok su linkovi na koje su spojeni krajnji uređaji podešeni kao pristupni linkovi (*access*), iako u našem slučaju koristimo samo jedan virtualni lan[42] (*vlan 1*). Ove postavke bit će nam poslije bitne prilikom konfiguracije i testiranja RSTP protokola jer da bi RSTP protokol mogao vršiti brzu konvergenciju, treba znati koji linkovi su mu *peer to peer* (spojeni na druge preklopnike), a koji su *edge* sučelja (spojeni na krajnje uređaje).

```
enable !ulazak u user mod
configure terminal !ulazak u exec mode
interface Gi0/0 !ulazak u konfiguraciju sučelja
switchport trunk encapsulation dot1q !specificiranje trunking
protokola
switchport mode trunk !definiranje sučelja kao trunk porta
exit !povratak u exec mode
interface Gi0/
switchport mode access !definicija sučelja kao access sučelja
switchport access vlan1 !definiranja vlan-a na sučelju
```

[42] Vrsta umrežavanja u kojem se domena prostiranja može nalaziti distribuirana na više preklopnika

```
interface vlan1 !konfiguracija SVI sučelja
ip address IPADRESA MASKA !podešavanje ip adrese preklopnika
no shutdown !uključenje sučelja
```

Kôd 4.2 Primjer konfiguracije *trunk* i *access* sučelja, na isti način konfigurirana su sva spojena sučelja na svim preklopnicima, ovisno o tipu uređaja koji je spojen na to sučelje

Svim virtualnim mašinama i usmjerniku podešena je mrežna povezivost, CentOS mašine sadrže sljedeću konfiguraciju sučelja:

```
TYPE=Ethernet
BOOTPROTO=static
IPADDR=10.0.0.1
NETMASK=255.255.255.0
GATEWAY=10.0.0.254
DEFROUTE=yes
DEVICE=ens33
ONBOOT=yes
```

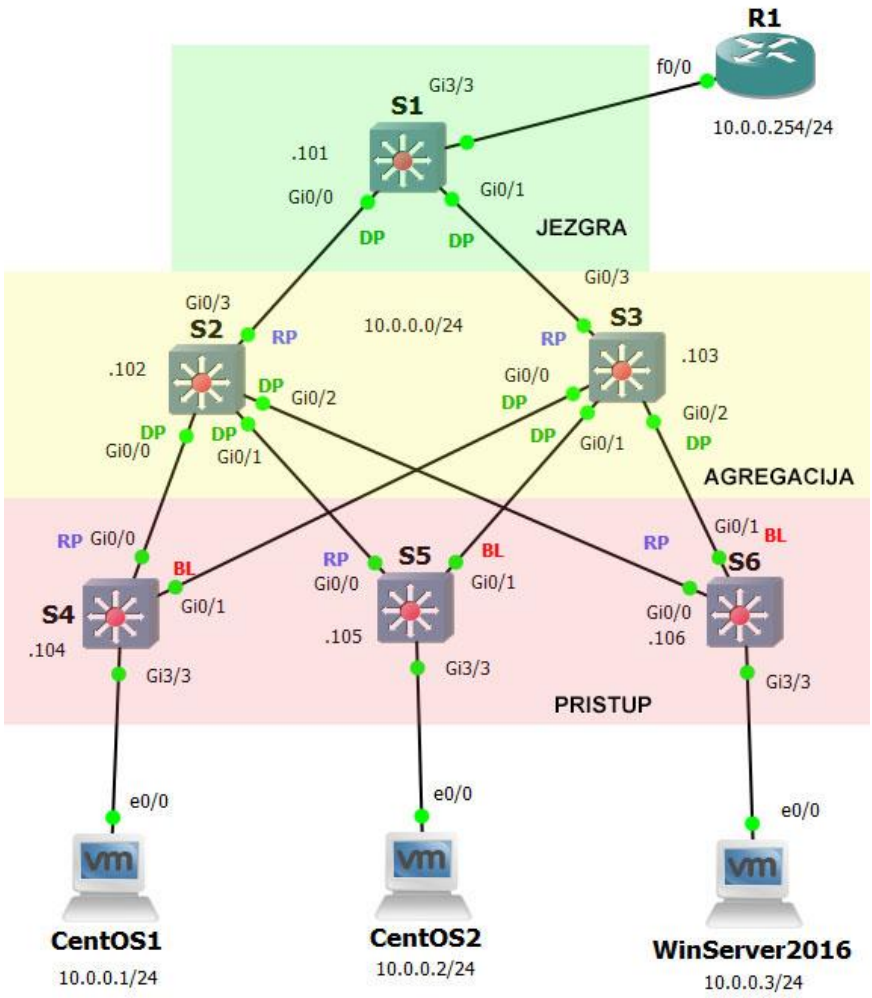
Kôd 4.3 Konfiguracija datoteke /etc/sysconfig/network-scripts/ifcfg-ens33 na CentOS1 virtualnoj mašini, ista konfiguracija podešena je i na drugoj mašini, jedina razlika je u IP adresi

Na isti način podešena je i mrežna adresa 10.0.0.3/24 na Windows Server 2016 virtualnoj mašini unutar GUI-ja.

Cisco Router na rubu mreže ima podešenu mrežnu adresu 10.0.0.254/24 sukladno sljedećoj naredbi:

```
enable !ulazak u user mod
configure terminal !ulazak u exec mode
interface Fa0/0 !ulazak u konfiguraciju sučelja
ip address 10.0.0.254 255.255.255.0 !ulazak u konfiguraciju
sučelja
no shut !uključivanje sučelja
```

Kôd 4.4 Konfiguracija R1 Cisco routera



Slika 4.1 Topologija tradicionalne mreže (GNS3 simulator)[43]

4.1.2. SDN mreža

Kao i u topologiji tradicionalne mreže, u SDN mreži implementirano je šest virtualnih preklopnika organiziranih u troslojnu mrežnu hijerarhiju s jednim jezgrenim, dva agregatna i tri pristupna preklopnika. Za razliku od tradicionalne mreže u kojoj sam implementirao preklopnike bazirane na Cisco IOS operativnom sustavu, u SDN topologiji implementirao sam OpenvSwitch SDN preklopnike (verzija 2.4.0). Navedeni preklopnici podržavaju Openflow protokol i putem njega su preko *southbound* sučelja spojeni sa SDN kontrolerom na posebnoj odvojenoj upravljačkoj mreži (192.168.0.0/24). Za potrebe ovog rada na

[43] Izvor: Vlastiti rad autora: 16.09.2018

southbound sučelju koristi se Openflow 1.0 protokol. To je, naime, jedina verzija protokola koju potpuno podržavaju i korišteni preklopnici i Aruba HP VAN SDN kontroler.[44]

Kao što se može vidjeti na slici 4.2, topologija je podijeljena u dvije mreže, produkcijsku i upravljačku. Produkcijska mreža identična je tradicionalnoj, osim po preklopnicima. Upravljačka mreža odvojena je na posebnom mrežnom segmentu (192.168.0.0/24) i unutar nje nalazi se jedan preklopnik (*GNS3 Ethernet Switch* bez dodatne konfiguracije), virtualna mašina sa softverom kontrolera (Debian Linux) te aplikacijski poslužitelj s instaliranim CENTos7 operativnim sustavom. Postavke aplikacijskog poslužitelja postavljene su na identičan način kao i postavke CentOS1 i CentOS2 mašina (opisano u prethodnom naslovu). Aplikacijski server ima, za razliku od drugih CentOS mašina, instaliran GUI zbog potrebe korištenje Firefox *web*-preglednika za ostvarivanje pristupa sučelju kontrolera.

Na SDN kontroleru podešena je statička IP adresa modifikacijom datoteke */etc/network/interfaces* na sljedeći način:

```
auto eth0
iface eth0 inet static
address 192.168.0.100
netmask 255.255.255.0
```

Kôd 4.5 Konfiguracija IP adrese na SDN kontroleru

Nakon toga izvršena je instalacija samog kontrolera korištenjem *dkpg* naredbe. Ovo će pokrenuti instalaciju softvera, baza podataka te svih potrebnih paketa.

```
sudo dpkg -i hp-sdn-ctl_2.8.8.0366_amd64.deb
service sdnc start
```

Kôd 4.6 Pokretanje instalacije i pokretanje kontrolera, prilikom instalacije korištene su zadane vjerodajnice Aruba HP VAN SDN kontrolera: sdn:skyline

Nakon instalacije kontroleru je moguće pristupiti na adresi *https://192.168.0.100:8443/sdn/ui* koristeći *web*-preglednik na aplikacijskom serveru. Koriste se iste vjerodajnice koje su korištene i prilikom instalacije.

Svaki od OpenvSwitch preklopnika unutar GNS3 simulatora dolazi konfiguriran s jednim upravljačkim sučeljem (eth0) te ostalim sučeljima u *bridge* načinu rada na koje spajamo

[44] http://docs.openvswitch.org/en/latest/topics/openflow/, 12.08.2018

krajnje uređaje, a prije svega potrebno je izvršiti inicijalnu konfiguraciju adrese upravljačkog sučelja:

```
auto eth0
iface eth0 inet static
address 192.168.0.1
netmask 255.255.255.0
```

Kôd 4.7 Konfiguracija adrese upravljačkog sučelja preklopnika S1; s obzirom na to da je OpenvSwitch baziran na Debian Linuxu, konfiguracija se ostvaruje modifikacijom datoteke /etc/network/ifaces. Na svakom OpenvSwitch preklopniku koristi se druga IP adresa, u skladu s brojem preklopnika

Nadalje, potrebno je konfigurirati korišteni *southbound* protokol i spojiti preklopnike s kontrolerom, a nakon toga u GUI sučelju SDN kontrolera bit će vidljiva ostvarena konekcija s preklopnikom. Slijedi konfiguracija OpenvSwitch preklopnika s komentarima, a na svakom preklopniku postavljene su iste opcije:

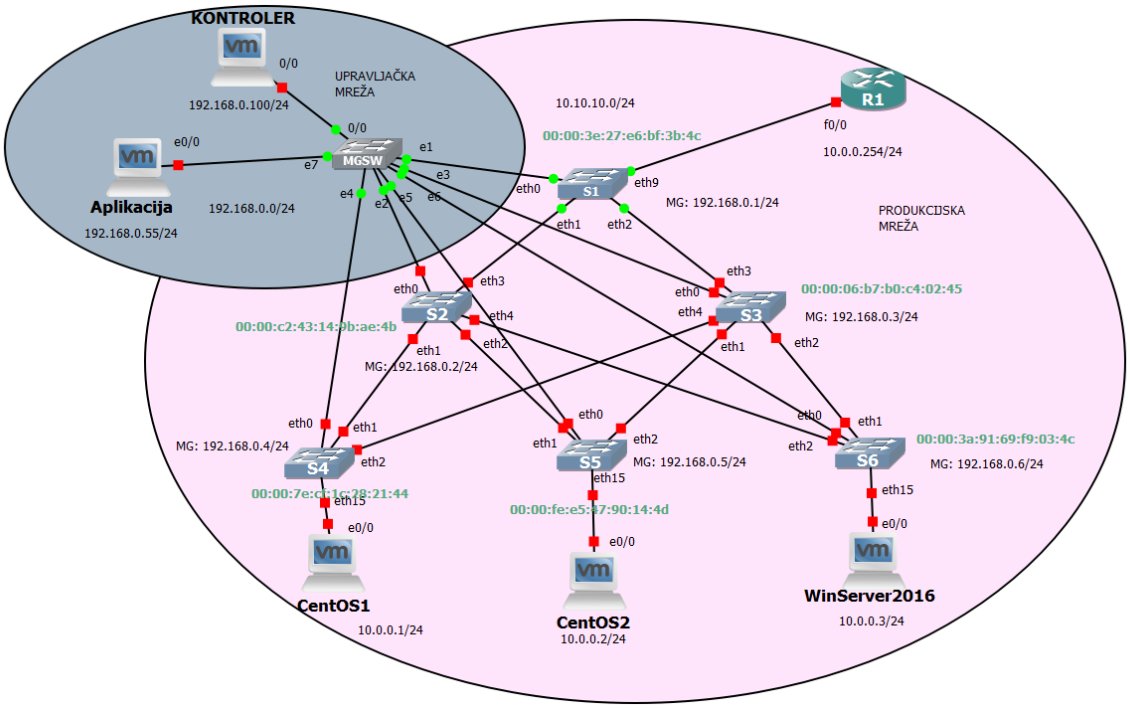
```
ovs-vsctl set bridge br0 protocols=OpenFlow10 #konfiguracija
korištenog southbound sučelja
ovs-vsctl set bridge br0 stp_enable=false #isključivanje STP
protokola
ovs-vsctl set-controller br0 tcp:192.168.0.100:6633
#ostvarivanje konekcije s kontrolerom
```

Kôd 4.8 Konfiguracija OpenvSwitch preklopnika

Data Path ID	*Address* ▲	Negotiated Version	Manufacturer	H/W Version	S/W Version
00:00:3e:27:e6:bf:3b:4c	192.168.0.1	1.0.0	Nicira, Inc.	Open vSwitch	2.4.0
00:00:c2:43:14:9b:ae:4b	192.168.0.2	1.0.0	Nicira, Inc.	Open vSwitch	2.4.0
00:00:06:b7:b0:c4:02:45	192.168.0.3	1.0.0	Nicira, Inc.	Open vSwitch	2.4.0
00:00:7e:cf:1c:28:21:44	192.168.0.4	1.0.0	Nicira, Inc.	Open vSwitch	2.4.0
00:00:fe:e5:47:90:14:4d	192.168.0.5	1.0.0	Nicira, Inc.	Open vSwitch	2.4.0
00:00:3a:91:69:f9:03:4c	192.168.0.6	1.0.0	Nicira, Inc.	Open vSwitch	2.4.0

Slika 4.2 Pogled na listu povezanih Openflow preklopnika na GUI sučelju SDN kontrolera[45]

[45] Izvor: Vlastiti rad autora, 15.08.2018



Slika 4.3 Topologija SDN mreže (GNS3 simulator)[46]

4.1.3. Vizualizacija tradicionalne mreže

Budući da za potrebe ovog rada koristim GNS3 mrežni simulator, omogućen mi je globalni pogled na tradicionalnu mrežu, kao što prikazuje Slika 4.1. No u produkcijskim uvjetima koji nisu virtualizirani, ovakav pogled na mrežu ne bi nam bio inicijalno dostupan. U takvim slučajevima potrebno je iscrtati fizičke i logičke dijagrame.

- Fizički dijagrami – prikazuju način na koji je oprema spojena u cjelinu (nazivi i brojevi spojenih sučelja, vrste spojeva, vrste i tipovi uređaja)
- Logički dijagrami – prikazuju logičke konstrukte koji definiraju kako će podaci putovati između mrežnih točaka; npr. IP adrese, MAC adrese, VLAN-ovi, korišteni protokoli itd.)

Za potrebe iscrtavanja dijagrama tradicionalne mreže pružaju nam se dvije opcije. Prva je opcija da ručnim pristupanjem sučeljima svih uređaja pažljivo čitamo konfiguracijske podatke. Kao pomoć pri čitanju podataka mrežnih uređaja možemo koristiti protokole za otkrivanje veza između njih kao što su Cisco Discovery Protocol (CDP) i Link Layer Discovery Protocol (LLDP) te dodatno ICMP protokol (ping i *traceroute* naredbe). CDP je

[46] Izvor: Vlastiti rad autora, 15.08.2018

vlasnički protokol od Cisca i funkcionira tako da uređaji koji ga koriste svakih šezdeset sekundi na posebnu multikast (engl. *Multicast*) adresu šalju podatke o sebi, što uključuje IP adrese, verziju operacijskih sustava i platformu, ime uređaja, tip uređaja, spojena sučelja i dr. Susjedni uređaji primaju te podatke i spremaju ih na određeno vrijeme (kod CDP-a to je vrijeme prema zadanom tri minute, ali ta se postavka može modificirati). Primjer ispisa CDP podataka s jezgrenog preklopnika S1 vidljiv je na sljedećoj slici, a u ovom slučaju korištene su naredbe *show cdp neighbours* (za dobivanje sažetih podataka o svim spojenim uređajima) te dodatno *show cdp neighbours detail* (za dobivanje detaljnijeg ispisa o svakom od spojenih uređaja, u ovom slučaju prikazan je detaljan ispis podataka o R1 uređaju):

```
S1#show cdp neighbors
Capability Codes: R - Router, T - Trans Bridge, B - Source Route Bridge
                  S - Switch, H - Host, I - IGMP, r - Repeater, P - Phone,
                  D - Remote, C - CVTA, M - Two-port Mac Relay

Device ID        Local Intrfce     Holdtme    Capability  Platform  Port ID
R1.vua-zavrsni.com
                 Gig 3/3           109              R    7206VXR   Fas 0/0
S3               Gig 0/1           135            R S I           Gig 0/3
S2               Gig 0/0           123            R S I           Gig 0/3

Total cdp entries displayed : 3
S1#show cdp neighbors detail
-------------------------
Device ID: R1.vua-zavrsni.com
Entry address(es):
  IP address: 10.0.0.254
Platform: Cisco 7206VXR,  Capabilities: Router
Interface: GigabitEthernet3/3,  Port ID (outgoing port): FastEthernet0/0
Holdtime : 101 sec

Version :
Cisco IOS Software, 7200 Software (C7200-ADVENTERPRISEK9-M), Version 15.2(4)M7, RELEASE SOFTWARE (fc2)
Technical Support: http://www.cisco.com/techsupport
Copyright (c) 1986-2014 by Cisco Systems, Inc.
Compiled Thu 25-Sep-14 10:36 by prod_rel_team

advertisement version: 2
Duplex: half
```

Slika 4.4 Primjer ispisa CDP naredbi na preklopniku S1 (tradicionalna mreža)[47]

Drugi je opcija da koristimo protokol LLDP koji nam može dati identične informacije, ali za razliku od CDP-a, nije vlasnički i formaliziran je kao standard od strane IEEE-a, a upravo zbog te činjenice koristi ga većina proizvođača mrežne opreme u svojim proizvodima. Prednost koju CDP ima nad LLDP protokolom jest da je CDP moguće koristiti i u L2 i u L3 okolinama, dok je LLDP moguće koristiti isključivo unutar L2 okoline.

Za otkrivanje je li neki uređaj živ te koliko se skokova nalazi od uređaja s kojeg testiramo koristili bismo ICMP protokol, konkretno naredbe *ping* i *traceroute*.

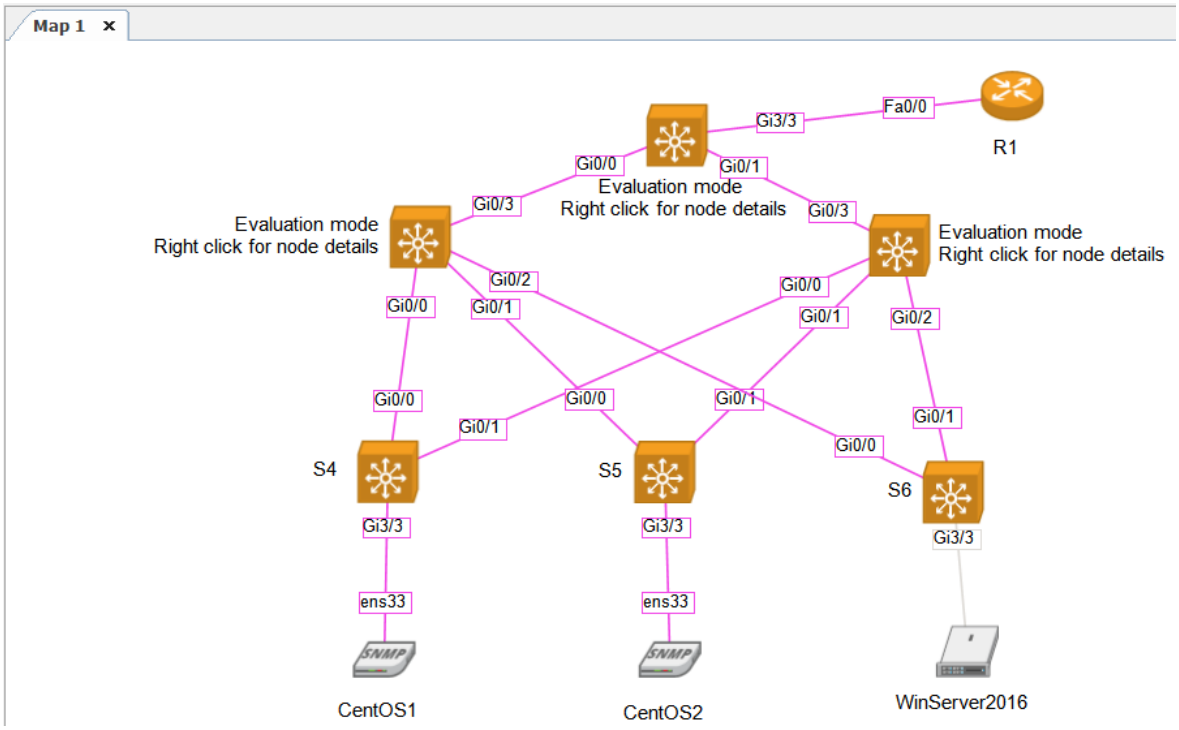
[47] Izvor: Vlastiti rad autora, 15.08.2018

Ovakav ručni način iscrtavanje topologije svakako je najsporiji, ali je bez obzira na to i dalje najpouzdanija metoda. Obično bismo sve podatke koje bismo na ovaj način prikupili popisali u tablicu te ih dodatno iscrtali kao mapu u alatima za izradu dijagrama kao što su *Microsoft Visio* ili *Edraw Max,* a sve ostale promjene također bi, naravno, trebalo redovito popisivati i mijenjati u dijagramima.

Drugi način prikupljanja ovih podataka je uz pomoć korištenje alata koji popisane korake rade automatski. Za potrebe ovog rada testirano je popularno rješenje *Network Topology Mapper* tvrtke *Solarwinds*. To je bilo jedino rješenje za koje sam uspio dobiti besplatno probno razdoblje za potrebe testiranja, obično su licencije za ovakve alate vrlo skupe i kreću se u rangu od nekoliko tisuća dolara. [48]

Nakon inicijalne registracije i preuzimanja, instalacija je izvršena na Windows Server 2016 mašini koja se nalazi na adresi 10.0.0.3. Network mapper uz ICMP uvelike ovisi o SNMP protokolu za čitanje podataka o mrežnoj topologiji (za dobivanje podataka o linkovima, SNMP u ovom slučaju također koristi podatke iz CDP tablica), tako da je trebalo izvršiti konfiguraciju koja je uključivala podešavanje SNMP *communityja* na svim uređajima u mreži. Rezultat skeniranja vidljiv je na Slika 4.5. Točnost dobivenog dijagrama u svakom je slučaju više nego zadovoljavajuća s obzirom na to da su svi uređaji i veze između njih očitane ispravno. Vrijeme trajanja skeniranja trajalo je 5 minuta i 28 sekundi (Slika 4.6). Ovaj rezultat svakako je bolji od ručnog popisivanja koje bi trajalo i do sat vremena na količinu ručnog rada koji je u tom slučaju potrebno izvršiti. S obzirom na visoku cijenu ovakvih alata, vjerojatno nisu isplativi mnogim manjim i srednje velikim organizacijama i upravo zbog te činjenice velika većina administratora oslanja se na prvu metodu ručnog dokumentiranja mrežnih dijagrama. Nadalje, čak i uza sve to, vrijeme od pet minuta koliko je trajalo generiranje naše testne topologije vrlo je dugo razdoblje za dobivanje uvida o trenutačnom stanju mreže, što je jedan od preduvjeta brze reakcije administratora pri detektiranju i uklanjanju kvarova. Nakon svake promjene u mreži potrebno je nanovo ručno pokrenuti skeniranje i generiranje dijagrama. U idućem poglavlju bit će prikazan način dobivanja trenutačne mrežne topologije u konzoli SDN kontrolera te izmjereno vrijeme potrebno da se isti dijagram očita.

[48]https://www.solarwinds.com/company/press-releases/solarwinds-network-topology-mapper-enhances-support-for-virtual-environments, 14.08.2018



Slika 4.5 Mapa testne tradicionalne mreže generirana u Solarwinds Network Topology Mapper alatu[49]

Slika 4.6 Vrijeme potrebno za automatsko učitavanje dijagrama sa slike 4.5[50]

4.1.4. Vizualizacija SDN mreže

Kako bi SDN kontroler mogao na pravilan način prosljeđivati pakete kroz mrežu zapisujući *flow* zapise u *flow* tablice preklopnika, on u svojoj bazi mora imati podatke o svim uređajima unutar SDN mreže te o svim vezama između njih, a ti se podaci moraju ažurirati u realnom vremenu.

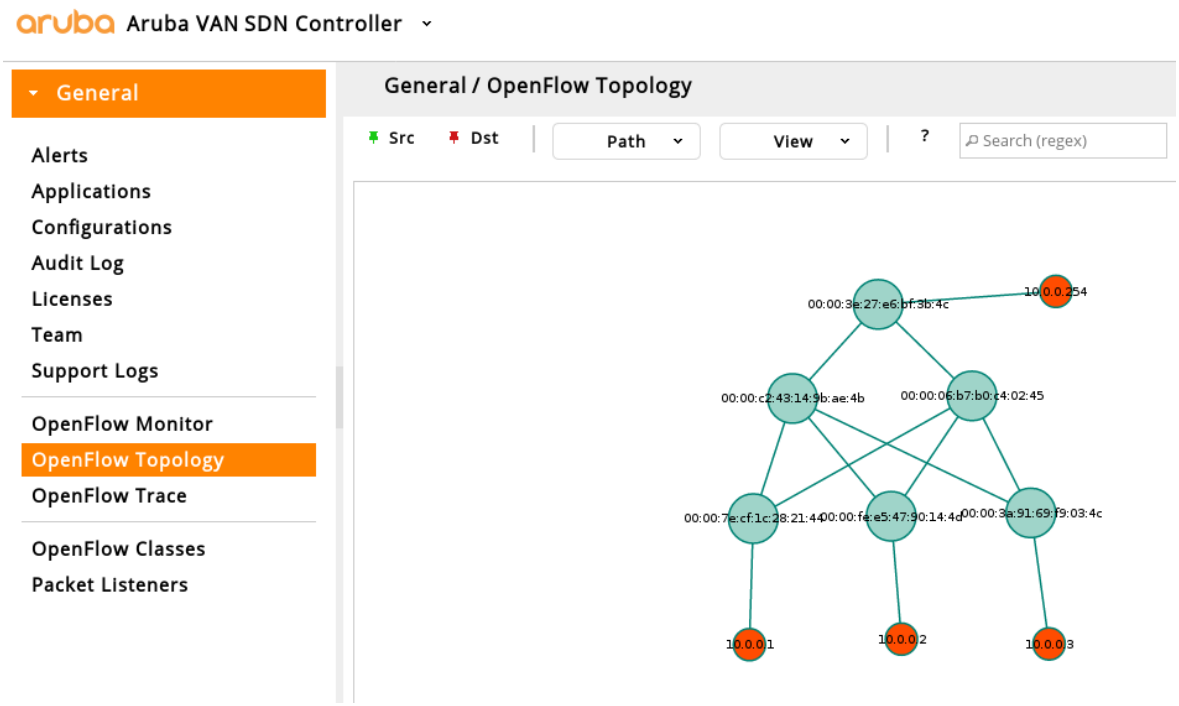
[49] Izvor: Vlastiti rad autora, 16.08.2018
[50] Izvor: Vlastiti rad autora, 16.08.2018

Kako bismo posve shvatili način na koji SDN kontroler dobiva informacije o uređajima u mreži, potrebno je razraditi korake koji se odvijaju na *southbound* sučelju prilikom i nakon uspostavljanja konekcije između kontrolera i preklopnika:

- Kontroler i preklopnik uspostavljaju konekciju putem TCP veze
- Vrši se razmjena OFPT_HELLO poruka između kontrolera i preklopnika, u ovom koraku definira se verzija Openflow protokola koja će se koristiti
- Kontroler šalje preklopniku OFPT_FEATURES_REQUEST poruku, a preklopnik odgovara s OFPT_FEATURES_REPLY. Unutar ove poruke nalaze se detalji o mogućnostima preklopnika, sadržavajući njegov DPID, 64-bitni jedinstveni identifikator (prvih 16 bita je ID instance, a ostalih 48 bita je MAC adresa preklopnika) te listu svih portova
- Nakon ovoga slijedi otkrivanje podataka o mreži, kontroler šalje OFPT_PACKET_OUT poruku s LLDP paketom na svaki od portova, LLDP paketi sadrže metapodatke o izvornom uređaju (ID preklopnika i ID porta). Unutar Openflow tablica svakog preklopnika postoji predefinirano pravilo da LLDP pakete koji ne dolaze od kontrolera šalju direktno kontroleru. U skladu s tim, ostali preklopnici kada prime poslane LLDP pakete, šalju ih prema kontroleru u vidu OFPT_PACKET_IN poruke. Sada kontroler na temelju metapodataka koji se nalaze u primljenom LLDP paketu može izračunati veze koje postoje između svih preklopnika i krajnjih uređaja. Na temelju tih podataka kontroler gradi topologiju SDN mreže. Ovakve poruke šalju se automatski u definiranom razdoblju (zadano svakih pet sekundi) kako bi kontroler imao potpunu sliku mreže u svakom trenutku.[51]

Nakon što su svi preklopnici spojeni s kontrolerom, u konzoli kontrolera na adresi *https://192.168.0.100:8443/sdn/ui* na linku Openflow Topology može se vidjeti trenutačna topologija SDN mreže, prikupljena na gore opisani način (Slika 4.7) Vrijeme ručnog osvježavanja dobivene topologije unutar grafičkog sučelja trajalo je 5 sekundi i 14 stotinki, vidljivo na Slika 4.8.

[51] D Hanas, O Mohamed, Efficient Topology Discovery in Software Defined Networks: Revised, 2017



Slika 4.7 Mapa testne SDN mreže generirana u *web*-sučelju Aruba VAN SDN kontrolera[52]

Slika 4.8 Vrijeme potrebno za automatsko iščitavanje dijagrama sa slike 4.7[53]

4.2. Upravljanje prometom u SDN okolini

Jedna od temeljnih prednosti koje SDN mrežna okolina nudi jest centralizirani i granularni pristup kontroli mrežnog prometa. SDN aplikacije putem *northbound* sučelja mogu proaktivno ili reaktivno utjecati na rad same mreže, ovisno o zahtjevima, zakrčenosti linkova ili zahtjevima korisnika, tj. aplikacija koje koriste.

[52] Izvor: Vlastiti rad autora, 16.08.2018
[53] Izvor: Vlastiti rad autora, 16.08.2018

U skladu s tim, aplikacije za kontrolu toka podataka mogu se svrstati u dvije temeljne skupine:

- Aplikacije kod kojih administratori ručno definiraju željeni tok podataka u mreži na proaktivan način – za potrebe ovog rada u *Bashu* je napisana upravo takva aplikacija, a rad aplikacije bit će u detalje objašnjen u nastavku
- Aplikacije koje same dinamički definiraju putanje prometa ovisno o stanju i zahtjevima mreže sukladno analizi, predikciji i iskorištenosti resursa u mreži – ovdje govorimo o kompleksnim aplikacijama kakve se koriste u mrežama većih organizacija i pružatelja usluga kao što su mreže Facebooka, Goldman Sachsa i ostalih.

U nastavku rada bit će prikazane mogućnosti proaktivnog upravljanja prometom u vidu interaktivne aplikacije pisane u *Bashu*. Aplikacija putem RESTful sučelja šalje JSON objekte prema kontroleru koji ih prevodi u Openflow poruke te ih implementira u tablice SDN preklopnika i tako utječe na samu putanju prometa. Nakon izvršenja skripte i rekonfiguracije putanji bit će izvršen test konvergencije mreže na način opisan na početku ovog poglavlja.

Budući da rad skripte ovisi o inicijalnoj autentifikaciji na kontroler te poslije o dohvatu i ispisu kompletnih podataka o uređajima u Openflow mreži, u idućem dijelu rada bit će zasebno objašnjene te funkcionalnosti, a nakon toga detaljno će biti opisan kod skripte, prikazane funkcionalnosti te testiran njezin rad.

4.2.1. Autentifikacija na kontroler

Kako bismo mogli prosljeđivati naredbe od strane aplikacije prema kontroleru te dohvaćati atribute i vrijednosti od strane kontrolera, potrebno je izvršiti inicijalnu autentifikaciju na kontroler putem REST sučelja, prilikom čega nam kontroler na korištenje daje token koji vrijedi 24 sata. Token se koristi za autentifikaciju kod svih daljnjih poziva. Sama autentifikacija obavlja se putem *Keystone Identity servisa,* zasebnog autentifikacijskog i autorizacijskog eksternog modula koji je implementiran kao dio arhitekture Aruba VAN SDN kontrolera. Dobiveni token spremit ćemo u zasebnu datoteku te se pozivati na nju prilikom svakog daljnjeg REST poziva.

Za dohvat i slanje podataka putem REST sučelja koristit ćemo aplikaciju Curl, a nju je potrebno instalirati na platformu na kojoj radimo. *Curl* za prijenos podataka može koristiti

niz mrežnih protokola kao što su FTP, SCP, TELNET i LDAP, no u našem slučaju te s obzirom na to da pristupamo REST sučelju kontrolera, koristit ćemo HTTPS protokol.

Prvo je potrebno dohvatiti autentifikacijske podatke korištenjem naredbe *curl*, zastavicom -k dopuštamo korištenje samopotpisanih certifikata, zastavicom -H definiramo tip sadržaja, što je u našem slučaju JSON objekt, dok zastavicom -d definiramo sam objekt, u ovom slučaju konkretan JSON kod koji kontroler očekuje na definiranom resursu, tj. na linku na kraju naredbe. Nakon izvršenja *curl* naredbe, za potrebe ispisa sadržaja JSON objekta u čitljivom obliku, čitav *output* filtriramo kroz json.tool u Pythonu.

```
curl -k -H 'Content-Type:application/json' -
d'{"login":{"user":"sdn","password":"skyline","domain":"sdn"}
}' https://192.168.0.100:8443/sdn/v2.0/auth | python -m
json.tool
```

Kôd 4.9 Dohvat i formatiranje autentifikacijskih podataka

```
[root@aplikacija ~]# curl -k -H 'Content-Type:application/json' -d '{"login":{"user":"sdn","passwor
d":"skyline","domain":"sdn"}}' https://192.168.0.100:8443/sdn/v2.0/auth | python -mjson.tool
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
100   346    0   286  100    60   1322    277 --:--:-- --:--:-- --:--:--  1330
{
    "record": {
        "domainId": "3706f39ba44e44f8be9e1d6fe21e85e5",
        "domainName": "sdn",
        "expiration": 1534632457000,
        "expirationDate": "2018-08-18 15-47-37 -0700",
        "roles": [
            "sdn-user",
            "sdn-admin"
        ],
        "token": "e6230412ebd24f4f93b6540927779e53",
        "userId": "4e1198e944714c5f93dd5b2ff8c4f123",
        "userName": "sdn"
    }
}
[root@aplikacija ~]#
```

Slika 4.9 *Output* koda 4.6[54]

Unutar dobivenog JSON objekta nalazi se nekoliko vrijednosti i njihovih atributa. Za potrebe autentifikacije potrebna nam je vrijednost atributa token, a nju ćemo izolirati uz pomoć naredbe *cut* te je spremiti u zasebnu datoteku imena *token*.

```
curl -k -H 'Content-Type:application/json' -d
'{"login":{"user":"sdn","password":"skyline","domain":"sdn"}}
' https://192.168.0.100:8443/sdn/v2.0/auth |

python -mjson.tool |
```

[54] Izvor: Vlastiti rad autora, 18.08.2018

```
grep token |
tr -d [:space:] |
tr -d '"' |
tr -d ',' |
cut -d ":" -f 2 > token
```

Kôd 4.10 Izrezivanje vrijednosti atributa token te njezino spremanje u datoteku token

Spremljeni token bit će nam potreban za autentifikaciju te za dobivanje autorizacije potrebnih za sva daljnja dohvaćanja, brisanja i slanja vrijednosti putem *northbound* sučelja.[55]

4.2.2. Dohvat podataka o uređajima na mreži

Za adekvatno upravljanje svim resursima koji se nalaze u Openflow mreži kontroler putem RESTful sučelja nudi aplikacijama pristup svakom od njih, što uključuje mogućnost dohvata informacija kao što su logovi, metrike i statistike, autentifikacijski i autorizacijski podaci, podaci o Openflow i krajnjim uređajima, podaci o licencijama, *clusteru* itd. Svakom od tih resursa može se pristupiti na zasebnom linku slanjem JSON objekta s parametrima koji su definirani u samoj dokumentaciji Aruba VAN SDN kontrolera. Dokumentaciji se može pristupiti na adresi *https://ADRESA:8443/api*.

Resursima je moguće pristupiti korištenjem neke od HTTP metoda:

- GET – dohvaćanje informacija o resursu ili resursima bez njegove modifikacije. Ti zahtjevi moraju biti sigurni (ne rade promjene na podacima) i idempotentni su, što znači da bez obzira na to koliko puta izvršili GET metodu, uvijek trebamo dobiti konzistentan rezultat [56]
- POST – zahtjev za izvršenjem promjene ili ažuriranja podataka resursa u skladu s poslanim objektom
- PUT – idempotentan zahtjev za stvaranjem novog resursa
- DELETE – zahtjev za brisanjem nekog od postojećih resursa.

Svaki od poslanih zahtjeva vratit će neki od statusnih kodova, ovisno o rezultatu:

- 1XX – informacijski
- 2XX – uspjeh

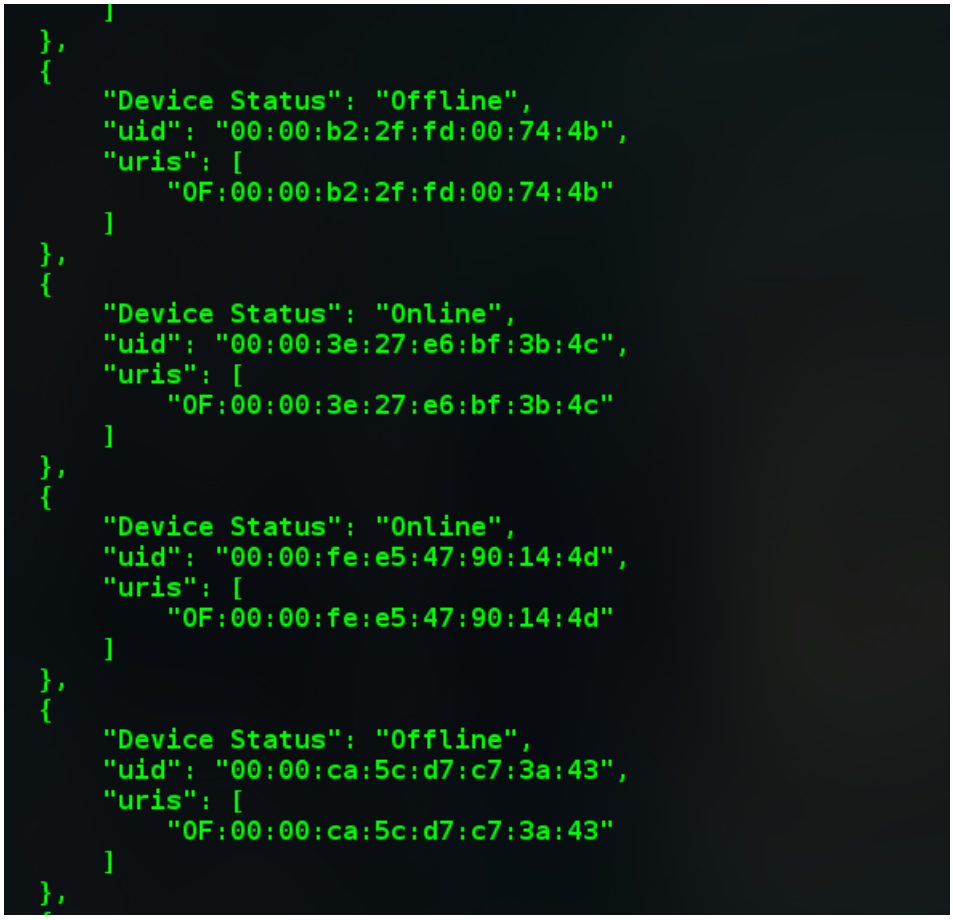
[55] D. Bombai; Complete, practical SDN and Openflow fundamentals; Udemy; (2017)
[56] https://www.restapitutorial.com/lessons/idempotency.html, 19.08.2018

- 3XX – preusmjerenje
- 4XX – greška na strani klijenta
- 5XX – greška na strani poslužitelja[57]

Kako bismo testirali pristup informacijama resursa/uređaja, zanima nas informacija o aktivnim i neaktivnim uređajima u SDN mreži, i koristit ćemo naredbu *curl* sa sljedećim parametrima:

```
curl -sk -H "X-Auth-Token:`cat token`" -H "Content-
Type:application/json"
https://192.168.0.100:8443/sdn/v2.0/net/devices | python -
mjson.tool
```

Kôd 4.11 Dohvat podataka o uređajima u mreži



Slika 4.10 Rezultat koda 4.8[58]

[57] https://spring.io/understanding/REST, 19.08.2018
[58] Izvor: Vlastiti rad autora, 19.08.2018

Budući da nam kontroler u ovom slučaju daje informacije o svim uređajima u mreži, oni koji su aktivni i oni koji nisu (korišteni u prošlosti), nas zanimaju samo trenutačni aktivni uređaji. Prethodni kod možemo spremiti kao zasebnu datoteku sa .sh ekstenzijom (u mojem slučaju *naprave.sh*), a nakon toga je možemo pokretati kao *bash* skriptu. Output prethodne naredbe možemo koristiti kao input za sljedeću naredbu korištenjem znaka *pipe* (|). Koristit ćemo alat jq. [59] Jq služi za filtriranje i rezanje JSON podataka. Koristit ćemo ga na sljedeći način:

```
./devices.sh | jq -r '.devices[] |select(."Device
Status"=="Online") | .uid'
```

Kôd 4.12 Filtriranje i ispis DPID aktivnih Openflow uređaja u mreži

Slika 4.11 Rezultat koda 4.9[60]

Kao ispis dobili smo popis DPID brojeva svih aktivnih Openflow uređaja u mreži, konkretno preklopnika, a iste podatke možemo vidjeti i unutar GUI kontrolera. No informacije koje smo uz pomoć naredbe *curl* dohvatili putem REST sučelja sada možemo koristiti kao varijable u skripti za upravljanje prometom.

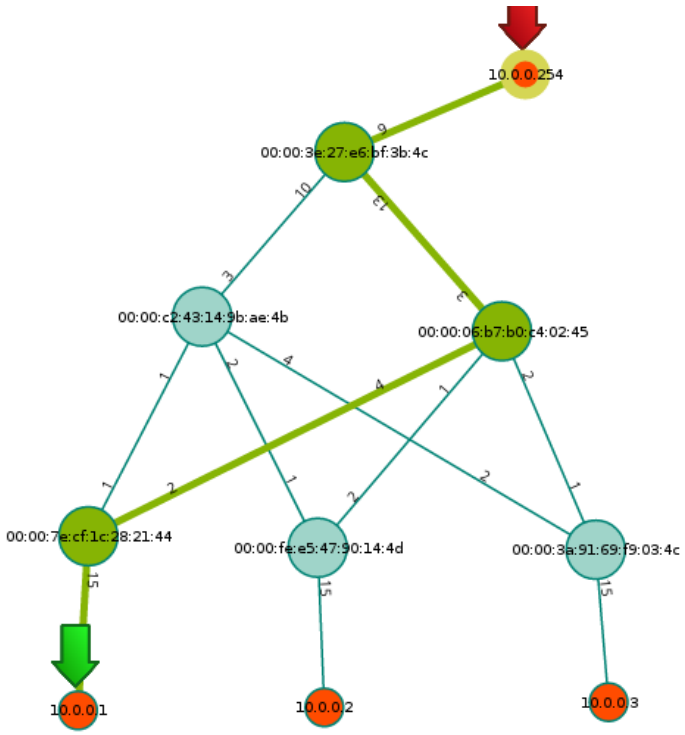
Data Path ID	Address	Negotiated Version	Manufacturer	H/W Version	S/W Version
00:00:3e:27:e6:bf:3b:4c	192.168.0.1	1.0.0	Nicira, Inc.	Open vSwitch	2.4.0
00:00:c2:43:14:9b:ae:4b	192.168.0.2	1.0.0	Nicira, Inc.	Open vSwitch	2.4.0
00:00:06:b7:b0:c4:02:45	192.168.0.3	1.0.0	Nicira, Inc.	Open vSwitch	2.4.0
00:00:7e:cf:1c:28:21:44	192.168.0.4	1.0.0	Nicira, Inc.	Open vSwitch	2.4.0
00:00:fe:e5:47:90:14:4d	192.168.0.5	1.0.0	Nicira, Inc.	Open vSwitch	2.4.0
00:00:3a:91:69:f9:03:4c	192.168.0.6	1.0.0	Nicira, Inc.	Open vSwitch	2.4.0

Slika 4.12 Provjera ispisa sa slike 4.11; kao što vidimo, rezultat dohvata odgovara onom što se može vidjeti unutar GUI kontrolera[61]

[59] https://stedolan.github.io/jq/, 20.08.2018
[60] Izvor: Vlastiti rad autora, 19.08.2018
[61] Izvor: Vlastiti rad autora, 19.08.2018

4.2.3. Skripta za upravljanje prometom

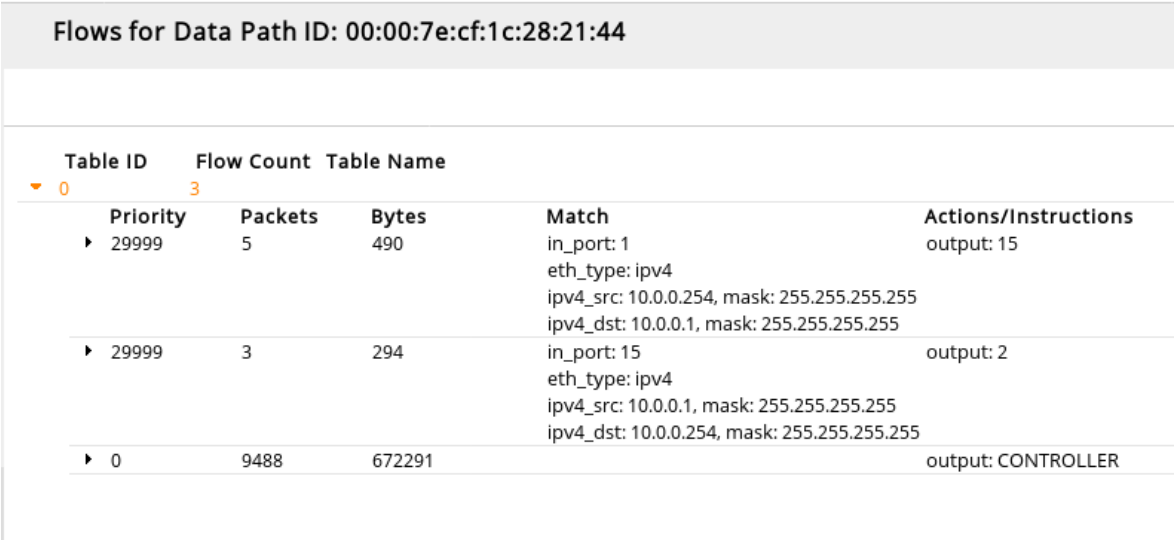
Aruba VAN SDN omogućava inicijalno usmjeravanje paketa u mreži koristeći internu aplikaciju *Path Daemon* koja izračunava najkraći put za sve pakete koje je proslijedio preklopnik uvidom u kompletnu topologiju mreže te u mogućnosti spojenih preklopnika i njihovih sučelja. Uvijek će biti odabran najkraći put u mreži prema broju skokova u i vrsti te brzini izlaznih sučelja preklopnika. Nakon što zaprimi dolaznu (*PACKET_IN*) poruku od preklopnika, Path Daemon na temelju izvršene kalkulacije vrši implementaciju *flow* zapisa u tablice preklopnika. U skladu s tim, u konzoli Aruba VAN SDN kontrolera možemo vidjeti trenutačno odabran najbolji put za svu komunikaciju između virtualne mašine CentOS1 na adresi 10.0.0.1 te usmjernika R1 na adresi 10.0.0.254:



Slika 4.13 Putanja prometa koju je definirala Path Daemon aplikacija[62]

Sav promet obavlja se putanjom S4-S3-S1. Detaljnim pregledom u stanje Openflow tablice preklopnika dobivamo na uvid implementirane zapise, a važno je primijetiti statističke informacije o broju paketa te o prioritetu zapisa implementiranog od strane kontrolera. Što je prioritet zapisa veći, on će biti preferiran prilikom odabira izlaznog sučelja.

[62] Izvor: Vlastiti rad autora, 19.08.2018



Slika 4.14 Detaljan prikaz zapisa u *flow* tablici preklopnika S4[63]

Za potrebe definiranja putanji prometa i same konvergencije mreže izrađena je interaktivna *bash* skripta koja će vršiti implementaciju *flow* zapisa s većim prioritetom od zadanog 29999 te u skladu s tim preusmjeravati sav dolazni promet od Centos01 virtualne mašine prema usmjerniku R1 putem preklopnika S4 na adresi 10.0.0.2. Kako bismo to postigli, skripta će morati vršiti sljedeće korake:

- Autentifikacija na kontroler
- Priprema privremenih datoteka
- Definiranje vremena *idle* i *hard timeouta* (vremena unutar kojeg će zapis biti valjan)
- Definiranje opcija filtriranja prometa (ovisno o OSI sloju na kojem se definiraju parametri)
- Prikupljanje, prikaz i odabir DPID brojeva Openflow preklopnika u mreži
- Definiranje izvorne i odredišne MAC adrese u slučaju L2 prometa, izvorne i odredište IP adrese u slučaju L3 prometa te izvorne i odredišne IP adrese i odredišnog porta u slučaju L4 prometa
- Definiranje aktivnog preklopnika
- Definiranje izlaznog sučelja na trenutačno aktivnom preklopniku
- Priprema JSON poziva u skladu s odabranim opcijama te njihovo spremanje u izvršnu datoteku

[63] Izvor: Vlastiti rad autora, 19.08.2018

- Slanje JSON objekata prema kontroleru izvršenjem skripte s pripremljenim JSON pozivima (nakon što su odabrani svi preklopnici i definirana sva izlazna sučelja).

U nastavku bit će prikazan kod skripte s komentarima. Za potrebe ovog rada i testiranja, prilikom izvršenja skripte bit će izabrano i prikazano filtriranje na temelju izvorne i odredišne IPv4 adrese te izlaznih sučelja, uz napomenu da je filtriranje moguće izvesti i na temelju izvornih i odredišnih MAC adresa te odlaznih TCP ili UDP portova.

Aplikacija se pokreće s aplikacijskog poslužitelja i ostvaruje komunikaciju sa SDN kontrolerom putem RESTful sučelja. Nakon što administrator definira na kojem sloju OSI modela želi raditi te unese tražene parametre, dobit će ispis svih preklopnika u mreži kao opciju, a nakon toga potrebno je za svaki od preklopnika definirati izlazno sučelje i tako definirati tok podataka od izvorišnog do odredišnog uređaja u mreži. Nakon što primi sve ulazne parametre od srane administratora, skripta ih šalje kontroleru kao JSON objekt. Kontroler nakon toga putem *southbound* sučelja, tj. Openflow protokola implementira pravilne *flow* zapise na svaki od SDN preklopnika u mreži.

```
#!/bin/bash

#Definiraj double quote
dqt='"'

#Uzmi token za daljnju autentifikaciju
source uzmi_token.sh

#Pripremi temp datoteke
rm -rf posalji_zapise.sh
touch posalji_zapise.sh
echo "#!/bin/bash" >> posalji_zapise.sh
chmod 755 posalji_zapise.sh

#Idle Timeout
echo -e "Definirajte Idle Timeout flow zapisa (u sekundama): \c"
read idle_timeout

#Hard Timeout
echo -e "Definirajte Hard Timeout flow zapisa (u sekundama): \c"
read hard_timeout

#Definiraj opcije filtriranja
opcije=("L2" "L3" "L4" "Prekid")

#Popuni array s napravama u SDN mreži
ARRAY=(`./devices.sh | jq -r '.devices[] |select(."Device Status"=="Online") | .uid'`)
echo

#Ponudi korisniku definirane opcije
echo "Odaberi mrežni sloj"
select opt in "${opcije[@]}"; do
case $opt in
```

Slika 4.15 Autentifikacija na kontroler, priprema pripremnih datoteka, definiranje vremena *idle* i *hard timeouta* te priprema opcija filtriranja i prikaz opcija filtriranja[64]

```
[root@aplikacija TE]# bash skripta.sh
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
100   346    0   286  100    60   1168    245 --:--:-- --:--:-- --:--:--  1176
token za autentifikaciju je uspjesno dohvacen, token je 174a0a7303e540e1968b2298a4be8dd2
Definirajte Idle Timeout flow zapisa (u sekundama): 600
Definirajte Hard Timeout flow zapisa (u sekundama): 600

Odaberi mrežni sloj
1) L2
2) L3
3) L4
4) Prekid
#?
```

Slika 4.16 Rezultat koda sa slike 4.15[65]

Tip paketa može se definirati na jednom od tri sloja OSI modela, točnije:

- Na drugom sloju OSI modela – uključuje izvorišnu i odredišnu MAC adresu
- Na trećem sloju OSI modela – uključuje izvorišnu i odredišnu IP adresu

[64] Izvor: Vlastiti rad autora, 19.08.2018
[65] Izvor: Vlastiti rad autora, 19.08.2018

- Na četvrtom sloju OSI modela – uključuje izvorišnu i odredišnu IP adresu te odredišno TCP ili UDP sučelje.

Za potrebe rada bit će odabrana opcija filtriranja na trećem sloju OSI modela, što uključuje izvornu i odredišnu IPv4 adresu te izlazne portove, vrijeme u kojem će *flow* zapisi biti valjani jest 10 minuta, tj. 600 sekundi, a što će biti i više nego dovoljno za potrebe testiranja. No u produkciji bismo ovo vrijeme povećali na dulje razdoblje ili bismo postavili vrijednost 0 za jednu i drugi varijablu, što bi značilo da se zapisi nikada ne brišu bez intervencije administratora. Važno je napomenuti da ovakva implementacija zapisa ima svojih nedostataka u otpornosti rekonfiguracije putanji u slučaju kvara u radu jednog od izlaznih sučelja bilo na kojem od preklopnika. U produkcijskoj mreži ovome bismo doskočili tako da bismo koristili neku od novijih verzija Openflow protokola (1.1 ili noviji) prilikom čega bismo koristili *Fast-Failover* mogućnost tako da bismo više sučelja postavili u zajedničku grupu sučelja te sav promet preusmjeravali u tu grupu umjesto na jedno sučelje. U slučaju kvara na jednom od fizičkih sučelja Openflow preklopnik detektirao bi taj kvar te automatski sav promet prosljeđivao na drugo aktivno sučelje u grupi. Tako bismo implementirali mehanizam redundancije sličan EtherChannel tehnologiji u tradicionalnom mrežama.

```
"L3")
        #Izvorna IP adresa
echo -e "Definirajte izvornu IP adresu: \c"
        read ipv4_src

        #Odredisna IP adresa
        echo -e "Definirajte odredisnu IP adresu: \c"
        read ipv4_dst

        #Definiraj JSON objekte i spremi ih u datoteke
        YODGOVOR="d"
        DILIN="d"
                while [ "$YODGOVOR" == "$DILIN" ]; do
                echo -n "Dodaj dpid novog preklopnika? (d/n): "
                read ODGOVOR
        YODGOVOR=`echo $ODGOVOR | tr [:upper:] [:lower:] | cut -c 1`
        if [ "$YODGOVOR" == "$DILIN" ]; then
                echo -e "Definiraj dpid preklopnika: \c"

                select opcija in "${ARRAY[@]}"; do
                dpid=$opcija
                echo "Izabrali ste $dpid"
        break
        done

                echo -e "Unesite izlazni port: \c"
                read izlaz

        #Postavi JSON objekte i pospremi ih u varijablu
        echo "{
                ${dqt}flow${dqt}: {
                ${dqt}priority${dqt}: 30000,
                ${dqt}idle_timeout${dqt}: $idle_timeout,
                ${dqt}hard_timeout${dqt}: $hard_timeout,
                        ${dqt}match${dqt}: [
                        {${dqt}eth_type${dqt}: ${dqt}ipv4${dqt}},
                        {${dqt}ipv4_src${dqt}: ${dqt}$ipv4_src${dqt}},
                        {${dqt}ipv4_dst${dqt}: ${dqt}$ipv4_dst${dqt}}
                                ],
                ${dqt}actions${dqt}: [{${dqt}output${dqt}: $izlaz}]
                }
        }" > $dpid

#Pošalji JSON objekt prema kontroleru
echo "curl -sk -H ${dqt}X-Auth-Token:$token${dqt} -X POST -H 'Content-Type:application/json' \
-d @$dpid https://192.168.0.100:8443/sdn/v2.0/of/datapaths/$dpid/flows" >> posalji_zapise.sh
fi
done
break
;;
```

Slika 4.17 Odabir izvorne i odredišne IPv4 adrese, odabira preklopnika na temelju DPID-a te određivanja izlaznog sučelja[66]

Nakon odabira načina filtriranja korisniku je postavljena opcija definiranja izvorne i odredišne IP adrese, nakon čega mu se na uvid daje lista svih aktivnih preklopnika u mreži identificiranih s njegovim *Datapath* identifikacijskim brojem (DPID).

Svaki preklopnik potrebno je odabrati odabirom opcije te definirati izlazno sučelje prometa. Na ovaj način može se definirati bilo koja od kombinacija putanji u mreži.

[66] Izvor: Vlastiti rad autora, 20.08.2018

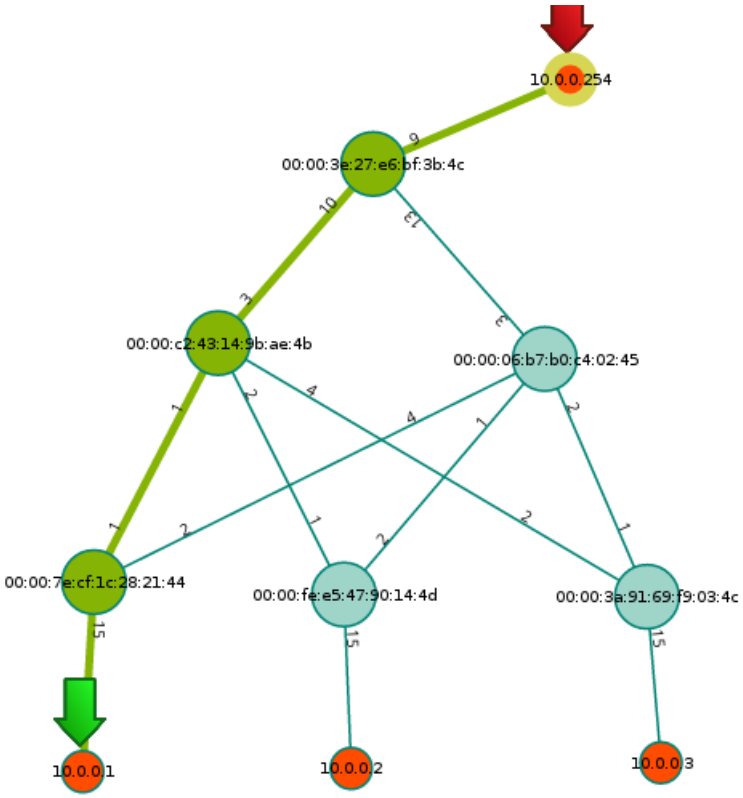
Skripta za svaki preklopnik sprema JSON objekt s definiranim poljima eth_type (sav ipv4 promet), ipv4_src (izvorišna adresa), ipv4_dst (odredišna adresa) te željeno izlazno sučelje u varijablu *dpid*, vrijednost varijable *dpid* poziva se unutar *curl* naredbe u skriptu posalji_zapise.sh, datoteku koja je bila pripremljena na početku, a na kraju skripte izvršava se posalji_zapise.sh sa svim *curl* pozivima. Tako se izvršava implementacija promjena u tablicama svih preklopnika u isto vrijeme.

```
Definirajte izvornu IP adresu: 10.0.0.1
Definirajte odredisnu IP adresu: 10.0.0.254
Dodaj dpid novog preklopnika? (d/n): d
Definiraj dpid preklopnika: 1) 00:00:c2:43:14:9b:ae:4b  4) 00:00:3e:27:e6:bf:3b:4c
2) 00:00:3a:91:69:f9:03:4c  5) 00:00:fe:e5:47:90:14:4d
3) 00:00:06:b7:b0:c4:02:45  6) 00:00:7e:cf:1c:28:21:44
#? 6
Izabrali ste 00:00:7e:cf:1c:28:21:44
Unesite izlazni port: 1
Dodaj dpid novog preklopnika? (d/n): d
Definiraj dpid preklopnika: 1) 00:00:c2:43:14:9b:ae:4b  4) 00:00:3e:27:e6:bf:3b:4c
2) 00:00:3a:91:69:f9:03:4c  5) 00:00:fe:e5:47:90:14:4d
3) 00:00:06:b7:b0:c4:02:45  6) 00:00:7e:cf:1c:28:21:44
#? 1
Izabrali ste 00:00:c2:43:14:9b:ae:4b
Unesite izlazni port: 3
Dodaj dpid novog preklopnika? (d/n): d
Definiraj dpid preklopnika: 1) 00:00:c2:43:14:9b:ae:4b  4) 00:00:3e:27:e6:bf:3b:4c
2) 00:00:3a:91:69:f9:03:4c  5) 00:00:fe:e5:47:90:14:4d
3) 00:00:06:b7:b0:c4:02:45  6) 00:00:7e:cf:1c:28:21:44
#? 4
Izabrali ste 00:00:3e:27:e6:bf:3b:4c
Unesite izlazni port: 9
```

Slika 4.18 Izvršenje koda sa slike 4.17, Definiranje izvorne i odredišne IP adrese, odabir svakog od preklopnika te definiranje izlaznih sučelja[67]

Nakon završetka rada skripte provjera putanje prometa u grafičkom sučelju Aruba VAN SDN kontrolera pokazuje sljedeći rezultat:

[67] Izvor: Vlastiti rad autora, 20.08.2018

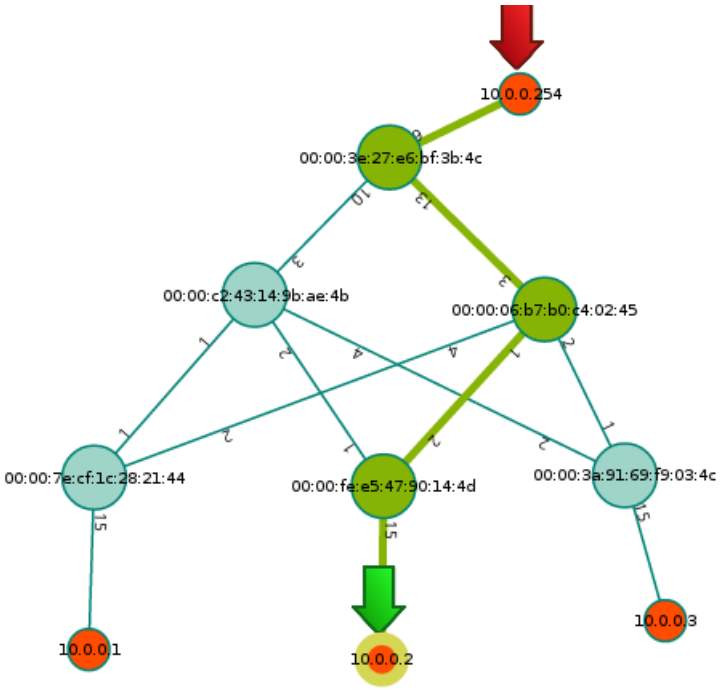


Slika 4.19 Putanja prometa koju je definirala skripta za upravljanje prometom, izvorište CentOS1 (10.0.0.1) i odredište usmjernik R1 (10.0.0.254)[68]

Sav promet od virtualne mašine CentOS1 (10.0.0.1) prema usmjerniku R1 (10.0.0.254) sada teče preko preklopnika S4-S2-S1, što je i bio cilj skripte.

Ako pak u grafičkom alatu Aruba VAN SDN kontrolera pregledamo putanju prometa od virtualne mašine CentOS2 (10.0.0.2) prema usmjerniku R (10.0.0.254), dobivamo originalni prikaz putanje prometa, čime dokazujemo da je upravljanje prometa u ovom slučaju granularno te se odnosi samo na pravila koja su definirana prilikom izvođenja skripte. Ovo je očekivano i ima smisla s obzirom na to da su izvođenjem skripte *flow* zapisi zapisani samo na preklopnike S4, S2 i S1.

[68] Izvor: Vlastiti rad autora, 20.08.2018



Slika 4.20 Prikaz putanje prometa kada je izvorište CentOS2 (10.0.0.2), a odredište usmjernik R1 (10.0.0.254)[69]

4.2.4. Pregled *flow* zapisa

Nakon izvršenja skripte *flow* zapisi zapisuju se u *flow* tablice preklopnika, a njih je moguće pregledavati unutar konzole kontrolera ili na samom preklopniku izvršenjem naredbe:

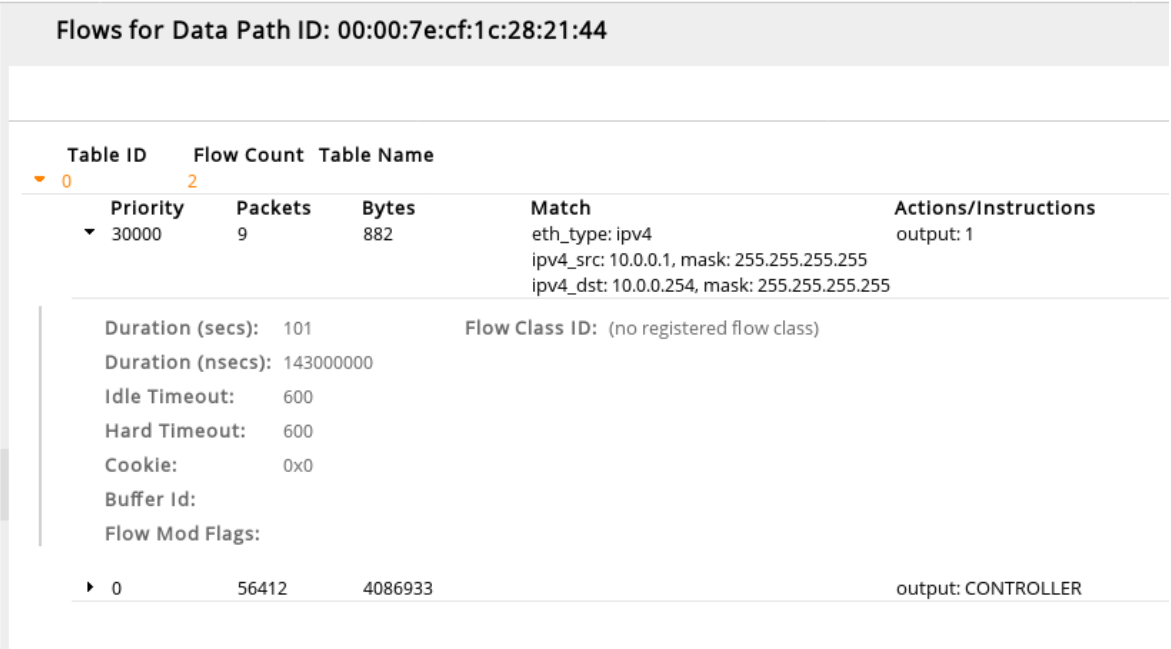
```
ovs-ofctl dump-flows br0
```

Kôd 4.10 Dohvat *flow* zapisa na preklopniku S4

```
/ # ovs-ofctl dump-flows br0
NXST_FLOW reply (xid=0x4):
 cookie=0x0, duration=150.847s, table=0, n_packets=9, n_bytes=882, idle_timeout=600, hard_timeout=600, idle_ag
e=142, priority=30000,ip,nw_src=10.0.0.1,nw_dst=10.0.0.254 actions=output:1
 cookie=0xfffe000000002328, duration=7.919s, table=0, n_packets=1, n_bytes=60, idle_timeout=8, hard_timeout=10
, idle_age=7, priority=28888,arp,in_port=1,dl_src=00:0c:29:af:92:59,dl_dst=00:0c:29:b4:cd:67 actions=output:15
 cookie=0xfffe000000002328, duration=7.915s, table=0, n_packets=0, n_bytes=0, idle_timeout=8, hard_timeout=10,
 idle_age=7, priority=28888,arp,in_port=15,dl_src=00:0c:29:b4:cd:67,dl_dst=00:0c:29:af:92:59 actions=output:1
 cookie=0xffff000000000000, duration=9226.772s, table=0, n_packets=56682, n_bytes=4106905, idle_age=0, priorit
y=0 actions=CONTROLLER:65509
/ #
```

Slika 4.21 Pregled implementiranog *flow* zapisa u konzoli preklopnika S4[70]

[69] Izvor: Vlastiti rad autora, 20.08.2018
[70] Izvor: Vlastiti rad autora, 20.08.2018



Slika 4.22 Prikaz implementiranog *flow* zapisa u konzoli kontrolera[71]

Flow zapisi slažu se prema prioritetima: zapis s većim prioritetom uvijek će imati prednost nad zapisom s nižim prioritetom, zapis s prioritetom 0 zove se *table-miss*, implementira ga kontroler i nalazi se na zadnjem mjestu u tablici te se primjenjuje za sve pakete koji se ne poklapaju sa zapisima većeg prioriteta u tablici. Svaki *flow* zapis jedinstveno je identificiran s prioritetom i *match* poljima i nije moguće implementirati dva identična *flow* zapisa s istim prioritetom jer u suprotnom dolazi do pogreške. Prilikom svakog poklapanja paketa s *match* poljima povećavaju se i brojači, broj prenesenih paketa te broj prenesenih bajtova.[72]

Na preklopniku S1 možemo vidjeti zapis u *flow* tablici s prioritetom 30000 te *match* poljima eth_type, ipv4_src i ipv4_dst. Ovaj *flow* zapis implementirala je naša skripta i ovime možemo potvrditi ispravnost konfiguracije. U skladu s postavljenim, preklopnici će u idućih 10 minuta sve pakete koji imaju postavljenu izvornu IP adresu 10.0.0.1 i odredišnu 10.0.0.254 usmjeravati na sučelje eth1. (Output 1). Na ostalim preklopnicima postoji identičan *flow* zapis, ali s drukčijim izlaznim sučeljima.

4.2.5. Konvergencija mreže

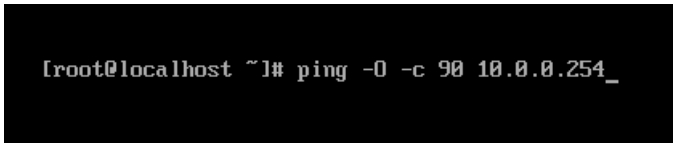
Konvergencija SDN mreže testirat će se slanjem ICMP paketa sa stanice CentOS1 korištenjem *echo-request* poruka prema usmjerniku R1. Kada usmjernik R1 zaprimi ICMP

[71] Izvor: Vlastiti rad autora, 20.08.2018
[72] J. Cassey; Introduction to SDN and Openflow;INE; (2015)

paket, natrag će poslati *echo-reply*, što predstavlja normalno stanje komunikacije između tih dvaju uređaja. ICMP paket slat će se svaku sekundu te će se poslati ukupno 90 paketa unutar vremena od minutu i trideset sekundi. U tom vremenu izvršit će se skripta za upravljanje prometom te će se sav promet preusmjeriti sa zadane putanje (S4-S3-S1) na putanju preko S2 preklopnika (S4-S2-S1).

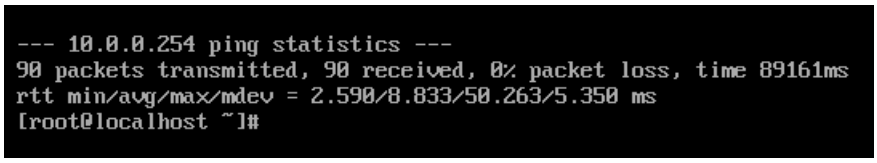
Na kraju testa zabilježit će se broj paketa izgubljenih na putu kroz mrežu, minimalna, prosječna i maksimalna latencija zabilježena prilikom slanja paketa te vrijeme unutar kojeg nije postojala komunikacija između uređaja koji su služili za testiranje.

Budući da skripta trenutačno implementira nove *flow* zapise na temelju kojih preklopnici rade odluke o preusmjeravanju paketa, očekuje se minimalan gubitak paketa i minimalna latencija te trenutačno preusmjeravanje u mreži.



Slika 4.23 Pokretanje ICMP testiranja[73]

Rezultat testiranja vidljiv je na Slika 4.24:



Slika 4.24 Rezultat testiranja s ICMP poruka u SDN mreži[74]

Od ukupno 90 poslanih ICMP *echo-request* poruka s virtualne mašine CentOS1 za svaku je zaprimljen odgovarajući *echo-reply* od strane usmjernika R1 u obliku *icmp-reply* poruke. Minimalno zabilježeno vrijeme odgovora je **2,590** milisekundi, prosječno vrijeme odgovora je **8,833** sekunde, dok je najveće zabilježeno vrijeme odgovora **50,263** milisekundi zabilježeno u vrijeme implementacije novih *flow* zapisa i konvergencije mrežnog prometa.

Budući da je konvergencija izvedena u instantnom vremenu, kao vrijeme nedostupnosti mreže uzet će se maksimalno vrijeme odziva u trenutačnu implementaciju novih zapisa, što je **50,263** milisekundi.

[73] Izvor: Vlastiti rad autora, 20.08.2018
[74] Izvor: Vlastiti rad autora, 20.08.2018

4.3. Upravljanje prometom u okolini tradicionalne LAN mreže

Upravljanje prometom u okolini tradicionalne mreže vršit će se manipulacijom STP i RSTP topologija. STP protokol služi sprečavanju petlji u redundantnim L2 mrežama definiranjem aktivne putanje u mreži za svaku od *broadcast* domena. RSTP je nadogradnja na STP standard koja je nastala kako bi se znatno ubrzala konvergencija mreže kod promjene topologije.

Redundancija u mreži predstavlja eliminaciju jedinstvene točke nastanka kvara kao što je uređaj ili dio uređaja (npr. fizičko sučelje) povezivanjem mreže tako da paketi uvijek mogu stići do destinacije na dva ili više načina. Redundancija predstavlja veliku prednost u ostvarivanju dostupnosti same mreže, ali zbog mogućnosti nastanka *broadcast* oluja zahtijeva korištenje STP protokola. STP jednostavno rečeno sprečava nastanak petlji, no način na koji to čini relativno je kompleksan te je za puno razumijevanje njegova mehanizama rada potrebno objasniti te mehanizme. Osim opisa standardnog STP protokola, bit će naznačene i razlike te dodatne mogućnosti koje su implementirane uvođenjem RSTP protokola koji je nastao zbog potreba za bržom konvergencijom u mreži prilikom promjene u topologiji mreže.

STP protokol i njegove varijante za potrebe komunikacije i određivanje stanja mreže i nužnih sljedećih koraka koriste *Bridge Protocol Data Units* (BPDU)[75] pakete koji se prema zadanom šalju na sve aktivne linkove svake dvije sekunde. Unutar BPDU poruka nalaze se ključni parametri na temelju kojih preklopnici mogu skupljati informacije stanju uređaja i linkova u mreži te mijenjati putanje prometa ovisno o tim informacijama. Unutar svakog BPDU paketa nalaze se informacije kao što su ID trenutačnog vršnog preklopnika, identifikator izvornog preklopnika, identifikator izvornog sučelja, vrijeme nastanka poruke, vrsta protokola, kumulativna cijena koštanja i dr.

Korištenjem STP-a i RSTP-a jedan preklopnik u mreži biva izabran kao vršni preklopnik te on postaje centralno referentno mjesto preko kojeg će teći sav promet. Kao vršni preklopnik bira se uređaj s najmanjim *Bridge ID-em*, a drugi preklopnici biraju najbolji put do njega. Prilikom kalkulacije najboljeg prometa gleda se redom sljedeće:

[75] BPDU – okvir unutar kojeg se prenose STP informacije

- Najmanja cijena linka ili linkova na putu do vršnog preklopnika – svako sučelje, ovisno o vrsti, nosi pripadajuću cijenu. 10Mbps link = 100, 100Mbps link = 19, 1Gbps link = 4 i 10Gbps link = 2. Sučelje s kumulativno najmanjom cijenom do vršnog preklopnika ima prednost i ono postaje vršno sučelje
- Najmanji Bridge ID – ID preklopnika u mreži sastoji se od prioriteta (2 bajta) i MAC adrese (6 bajtova) – za potrebe rada bit će izvršena manipulacija vrijednosti ovog polja s obzirom na to da svi linkovi u mreži imaju istu cijenu linkova (4)
- Najmanja brojčana oznaka sučelja – npr. Fa0/0 je manje od Fa0/1.

Nakon konvergencije mreže u kojoj se koristi STP protokol sučelja na preklopnicima postavljaju se u jedno od četiri moguća stanja:

- Vršno sučelje (*Root port*) – sučelje s najkraćim putem do vršnog preklopnika
- Odabrano sučelje (*Designated port*) – aktivno sučelje koje prima pakete i prosljeđuje ih prema vršnom preklopniku
- Blokirano sučelje (*Blocked port*) – neaktivno blokirano sučelje koje ne prosljeđuje promet
- Neaktivno sučelje (*Disabled port*) – ugašeno sučelje.

Korištenjem RSTP protokola, s druge strane, sučelja se postavljaju u jedno od pet mogućih stanja:

- Vršno sučelje (*Root port*) – sučelje s najkraćim putem do vršnog preklopnika
- Odabrano sučelje (*Designated port*) – aktivno sučelje koje prima pakete i prosljeđuje ih prema vršnom preklopniku
- Alternativno sučelje (*Alternate port*) – najbolji alternativni put do vršnog preklopnika, koristi se za brzu konvergenciju u slučaju da se na isto sučelje zaprimi superiorna BPDU poruka (manja cijena linkova ili manji *Bridge ID*)
- Rezervno sučelje (*Backup port*) – *backup* odabranog sučelja na drugom mrežnom segmentu unutar iste kolizijske domene (ovo se može dogoditi jedino ako je jedan segment mreže spojen na HUB, što se u modernim implementacijama rijetko ili gotovo nikada se koristi u praksi)
- Neaktivno sučelje (*Disabled port*) – ugašeno sučelje.

Prilikom konvergencije mreže i inicijalizacije konekcije svako od sučelja na preklopnicima prolazi kroz jedno od sljedećih stanja. U tablici je vidljiva usporedba stanja sučelja kod STP protokola te kod RSTP protokola:

STP		**RSTP**	
Blocking	Ne šalje i ne prima okvire, sluša samo na BPDU poruke, *Max age* (20 s)	Discarding	Ne šalje i ne prima okvire, sluša samo na BPDU poruke
Listening	Procesuira BPDU poruke i čeka na dodatne informacije koje će ga vratiti u prethodno ili postaviti u iduće *Forward delay* (15 s)	Discarding	Ne šalje i ne prima okvire, sluša samo na BPDU poruke
Learning	Prima i šalje BPDU poruke te uči MAC adrese, ne šalje okvire *Forward delay* (15 s)	Learning	Prima i šalje BPDU poruke te uči MAC adrese, ne šalje okvire
Forwarding	Prima i šalje sve podatke	Forwarding	Prima i šalje sve podatke
Disable	Onemogućeno sučelje	Discarding	Onemogućeno sučelje

Tablica 4.1 Usporedba mogućih stanja sučelja kod STP i RSTP protokola

Kod
rekonfiguracije STP protokola moguće su dvije vrste prekida veze:

- Direktne – kada se prekid dogodi na jednom od direktno spojenih sučelja, bira se novo vršno sučelje i sva sučelja prolaze kroz *listening* i *learning* stanja, nema *blocking* stanja, vrijeme rekonfiguracije je 2 x *Forward Delay*
- Indirektne – kada se prekid dogodi na nekom drugom linku koji nije iz perspektive sadašnjeg preklopnika direktno spojen na njega, kod takve rekonfiguracije sadašnji preklopnik gubi konekciju s vršnim preklopnikom i sebe proglašava novim vršnim preklopnikom te šalje BPDU poruke koje su iz perspektive susjednog preklopnika inferiorne. Čeka se da blokirani link „zastari“ tako da prođe kroz *Max age* stanje

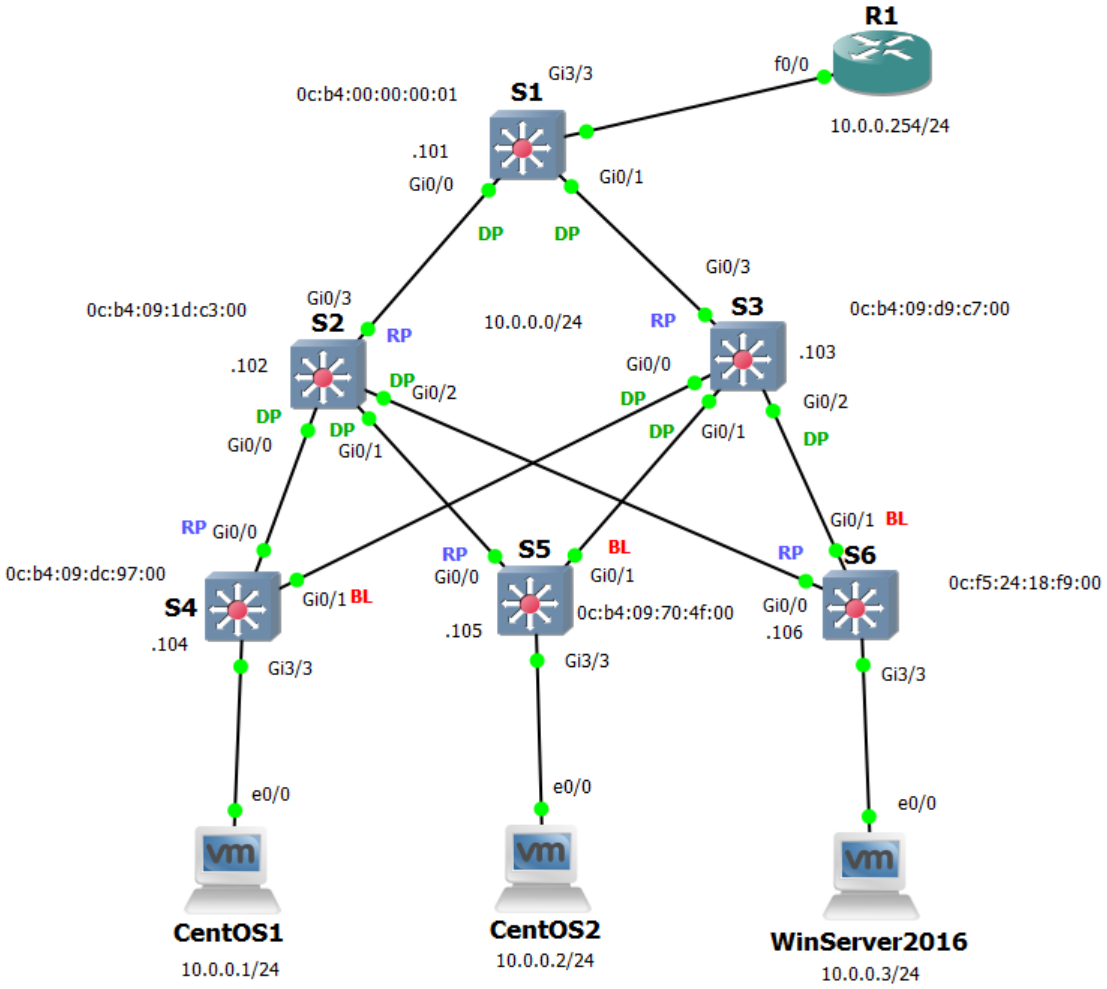
nakon kojeg link počinje dobivati bolje informacije te prelazi u željeno stanje. Vrijeme je *Max age* (30 s) + 2 x *Forward Delay* (15 s)[76]

STP protokol mora prolaziti kroz *Max Age* i *Forwarding delay* stanja prilikom rekonfiguracije mreže, što stvara relativno dugo vrijeme konvergencije od minimalno 30 sekundi.

RSTP protokol ne koristi *forwarding* i *Max age* brojača, za razliku od STP protokola kod kojeg se odluka o topologiji mreže obavlja isključivo na temelju BPDU poruka koje dolaze od vršnog preklopnika. RSTP protokol ponaša se slično kao i L3 *routing* protokoli. Svaki preklopnik komunicira na svim linkovima sa svojim susjedom, ali se topologijska tablica ne čuva lokalno već se uspostavljaju mehanizmi brzog oporavka korištenjem alternativnih *root* portova. Ako preklopnik primijeti na nekom od portova superioran BPDU, automatski stavlja sve portove u *discarding* stanje, šalje *agreement* prema preklopniku koji je odaslao BPDU prouku i instantno postavlja sučelje kao vršno. Sinkronizacijske se poruke tako šire kroz mrežu u obliku vala, dok sva sučelja na svim preklopnicima nisu sinkronizirana, a konvergencija mreže izvodi se u maksimalnom roku od 3 x *Hello* poruke (3x2 sekunde prema zadanom) kod indirektnog prekida ili u manje od 1 sekunde kod direktnog prekida te kod ručne promjene u topologiji mreže.

U testnoj topologiji prema zadanom koristi se RSTP protokol, a preklopnik S1 izabran je kao vršni preklopnik zbog činjenice što ima najmanju MAC adresu. Ako virtualna mašina CentOS1 šalje promet prema R1 usmjerniku, sva komunikacija ići će preko preklopnika S2 jer je sučelje Gi0/1 na mašini CentOS1 u blokiranom stanju.

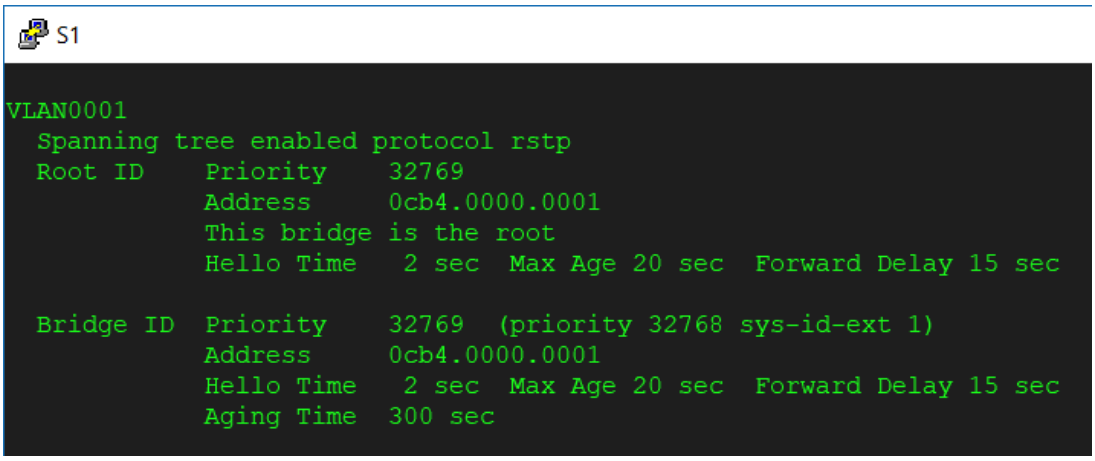
[76] http://blog.ine.com/2009/03/07/understanding-stp-convergence-part-i/, 29.08.2018



Slika 4.25 Testna topologija tradicionalne mreže[77]

Detaljnije informacije u konzoli usmjernika S1:

[77] Izvor: vlastiti rad autora, 29.08.2018



Slika 4.26 STP podaci na preklopniku S1; preklopnik S1 postao je vršni preklopnik u skladu s činjenicom da ima najmanju MAC adresu u mreži[78]

Prioritet svih preklopnika u mreži je prema zadanom isti (32768 + VLAN ID = 32769).

Kako bi za potrebe testiranja uvjeti u našoj mreži postali identični onima u SDN topologiji, valja podesiti mrežu tako da sav promet prolazi kroz preklopnik S3. Nakon toga valja izvršiti testove vremena konvergencije i broja dobivenih i izgubljenih paketa. Prva konfiguracija i testiranje izvršit će se korištenjem STP protokola, a s obzirom na ručnu rekonfiguraciju putanji rekonfiguracijom trenutačnog vršnog preklopnika, očekuje se vrijeme prestanka rada mreže od 2 x *Forward Delay*, što je 30 sekundi.

Prilikom testiranja RSTP protokola očekivano vrijeme konvergencije manje je od dvije sekunde zbog postavljenih alternativnih vršnih portova, instantne sinkronizacije korištenjem pregovora i dogovora između preklopnika na svakom linku te relativno malog broja uređaja u mreži.

4.3.1. Konfiguracija STP protokola i upravljanje tokom podataka

STP protokol tvornički je uključen na svim preklopnicima u mreži. S obzirom na zadani mod rada RSTP-a, za potrebe testiranja potrebno je prvo aktivirati standardni STP protokol:

```
spanning-tree mode pvst
```

Kôd 4.13 Konfiguracija Per VLAN STP protokola

Naknadno, prilikom vraćanja konfiguracije RSTP protokola, dovoljno ga je uključiti korištenjem iste naredbe s drugom opcijom na svakom od preklopnika:

[78] Izvor: vlastiti rad autora, 29.08.2018

```
Spanning-tree mode rapid-pvst
```

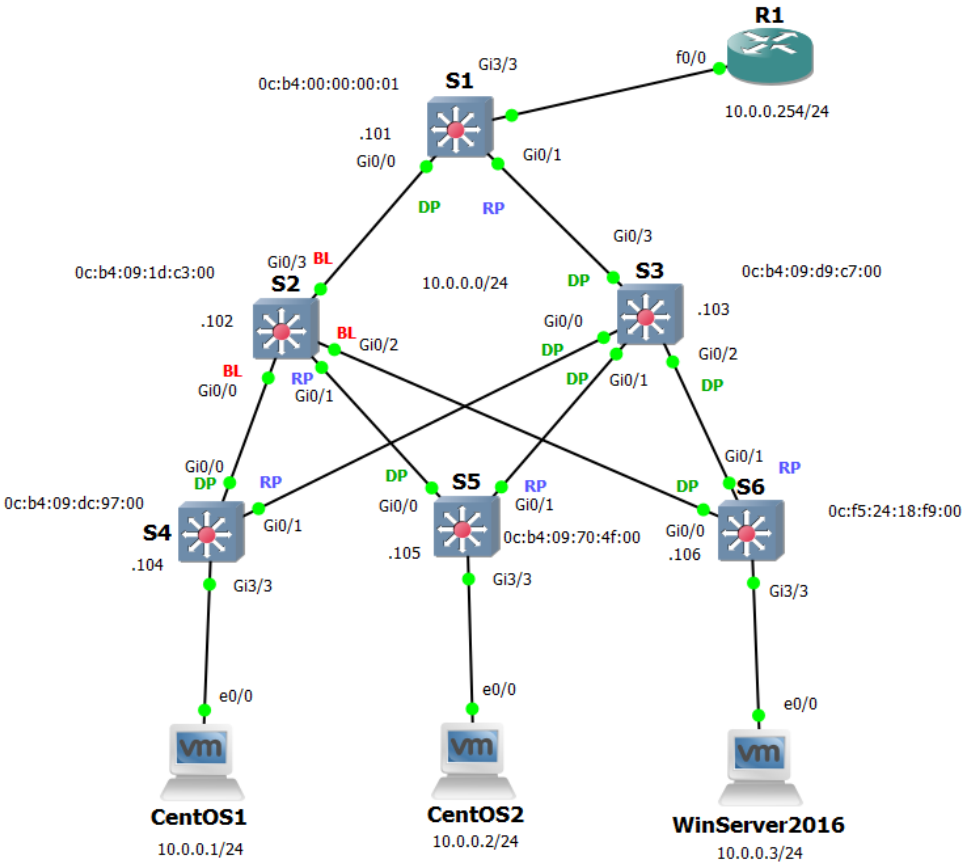
Kôd 4.14 Konfiguracija prioriteta preklopnika

Naredbu treba izvršiti na svim preklopnicima u mreži. Nakon toga na preklopniku S3 treba podesiti manji prioritet, što će učiniti da sav promet putuje preko njega:

```
Spanning-tree vlan 1 priority 8192
```

Kôd 4.15 Konfiguracija prioriteta preklopnika

Kao prioritet uzima se bilo koji broj manji od 32768 i veći od 4096. Broj mora biti inkrementiran s 4096 te veći ili jednak broju 4096. Za potrebe rada uzet je broj 8192 (2 x 4096), što će postaviti ID preklopnika na 8193, nakon što uzmemo u obzir i broj VLAN-a (1). Budući da svi drugi prekopnici u mreži imaju prioritet 32769, preklopnik S3 postat će automatski vršni preklopnik i svi će preklopnici slati promet preko njega.



Slika 4.27 Stanje svih sučelja nakon što je S3 postao izvorni preklopnik[79]

[79] Izvor: vlastiti rad autora, 29.08.2018

S3

```
VLAN0001
  Spanning tree enabled protocol rstp
  Root ID     Priority    8193
              Address     0cb4.09d9.c700
              This bridge is the root
              Hello Time   2 sec  Max Age 20 sec  Forward Delay 15 sec

  Bridge ID   Priority    8193   (priority 8192 sys-id-ext 1)
              Address     0cb4.09d9.c700
              Hello Time   2 sec  Max Age 20 sec  Forward Delay 15 sec
              Aging Time  300 sec
```

Slika 4.28 Stanje ispisa STP konfiguracije na S3 preklopniku[80]

4.3.2. Konvergencija mreže

Sukladno prijašnjoj konfiguraciji, vršni preklopnik postao je S3. Za potrebe testiranja konvergencije mreže izvode se isti testovi koji su bili korišteni i kod SDN topologije, i to u dva navrata, prvo uz korištenje STP protokola, a zatim uz korištenje RSTP protokola.

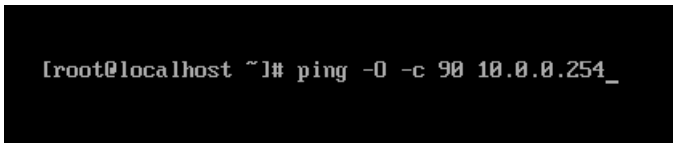
Konvergencija mreže testirat će se slanjem ICMP paketa sa stanice CentOS1 korištenjem *echo-request* poruka prema usmjerniku R1. Kada usmjernik R1 zaprimi ICMP paket, natrag će poslati *icmp echo-reply*, što predstavlja normalno stanje komunikacije između tih dvaju uređaja. ICMP paket slat će se svaku sekundu te će se poslati ukupno 90 paketa unutar vremena od minutu i trideset sekundi. U tom vremenu izvršit će se rekonfiguracija vršnog preklopnika tako da će novi vršni preklopnik postati S1, što će izazvati konvergenciju prometa.

U skladu s trenutačnom konfiguracijom komunikacija između virtualne mašine CentOS1 i usmjernika kreće se putem S4-S3-S1-R1.

Kod STP protokola očekivano vrijeme konvergencije je 2 x *Forward Delay* (15 sekundi) + vrijeme slanja BPDU poruka (2 sekunde), čime dolazimo do brojke od približno 32 sekunde.

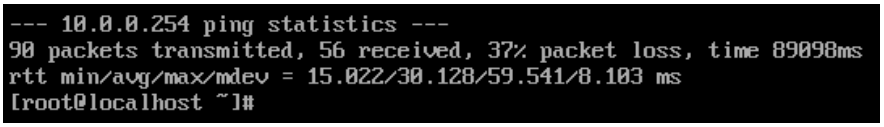
Na kraju testa zabilježit će se broj paketa izgubljenih na putu kroz mrežu te vrijeme unutar kojeg nije postojala komunikacija između uređaja koji su služili za testiranje.

[80] Izvor: vlastiti rad autora, 29.08.2018



Slika 4.29 Slanje *echo-reply* poruke od CentOS1 prema usmjerniku R1[81]

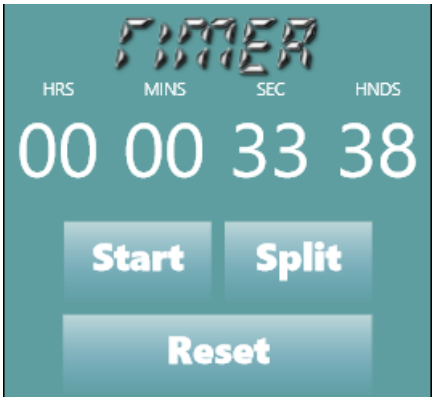
Prilikom testiranja na preklopniku S1 smanjuje se prioritet na 8192, što izaziva stanje da preklopniku R1 postaje vršni preklopnik i rekonfiguraciju mreže. Niti jedan paketi više se ne šalje putem S4-S3-S1-R1, nego putem S4-S2-S1-R1. Nakon rekonfiguracije rezultati *ping* naredbe koja se izvršavala u pozadini na virtualnoj mašini CentOS1 su sljedeći:



Slika 4.30 Rezultat testiranja s ICMP poruka u tradicionalnoj mreži (STP)[82]

Od ukupno **90** poslanih *echo-request* poruka primljene su **56** poruke, **34** poruke nisu stigle na destinaciju, što je gubitak od **37 %**. Minimalno vrijeme odgovora je **15,022** ms, prosječno vrijeme odgovora je **30,128** milisekundi, a maksimalno vrijeme odgovora paketa je **59,541** milisekundi.

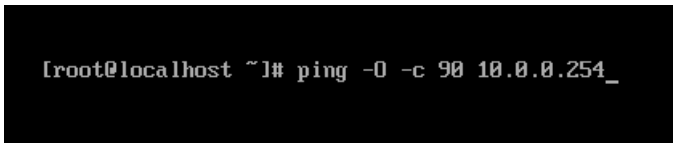
Vrijeme trajanja konvergencije mreže izvršeno je pokretanjem brojača od vremena prve pa do vremena zadnje propuštene *echo-reply* poruke. Izmjereno vrijeme od **33 sekunde i 38 stotinki** odgovara očekivanom rezultatu od približno 32 sekunde.



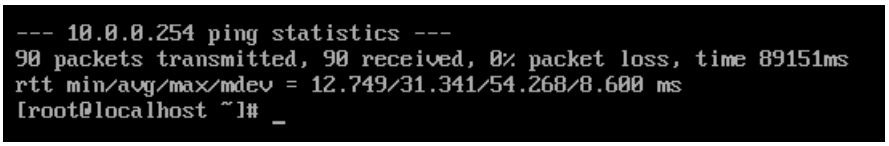
Slika 4.31 Vrijeme trajanja konvergencije mreže u STP topologiji[83]

[81] Izvor: vlastiti rad autora, 29.08.2018
[82] Izvor: vlastiti rad autora, 29.08.2018
[83] Izvor: vlastiti rad autora, 29.08.2018

Za potrebe testiranja konvergencije u RSTP mreži prvo se obavlja konfiguracija RSTP protokola na svim preklopnicima te se pokreću isti testovi kao i prije. Očekivano vrijeme konvergencije je unutar šest sekundi, iako bi se, zbog činjenice da vršimo konfiguraciju mreže bez prekida u radu nekog od *portova*, konvergencija trebala dogoditi u manje od dvije sekunde.



Slika 4.9 Slanje *echo-reply* poruke od CentOS1 prema usmjerniku R1[84]



Slika 4.32 Rezultat ICMP testiranja u tradicionalnoj mreži (RSTP)[85]

Od ukupno 90 poslanih ICMP *echo-request* poruka s virtualne mašine CentOS1 za svaku je zaprimljen odgovarajući *echo-reply* od strane usmjernika R1 u obliku *icmp-reply* poruke, minimalno zabilježeno vrijeme odgovora je **12,749** milisekundi, prosječno vrijeme odgovora je **31,341** sekunde, dok je najveće zabilježeno vrijeme odgovora **54,268** milisekundi.

Budući da je konvergencija mreže korištenjem RSTP protokola izvedena instantno kao i u slučaju SDN topologije, vrijeme nedostupnosti mreže uzet će se maksimalno vrijeme odziva, što je **54.268** milisekundi.

4.4. Analiza rezultata

Testiranjem vremena konvergencije u SDN mreži te u tradicionalnoj mreži korištenjem STP protokola te RSTP protokola dobili smo sljedeće rezultate:

	SDN topologija	Tradicionalna topologija (STP)	Tradicionalna topologija (RSTP)
Minimalno vrijeme odgovora	2,590 ms	15,022 ms	12,749 ms

[84] Izvor: vlastiti rad autora, 29.08.2018
[85] Izvor: vlastiti rad autora, 29.08.2018

Prosječno vrijeme odgovora	8,833 ms	30,128 ms	31,341 ms
Maksimalno vrijeme odgovora	50,263 ms	59,541 ms	54,268 ms
Vrijeme nedostupnosti mreže	50,263 ms	33,38 s	54,268 ms
Broz izgubljenih paketa (od 90)	0	34	0
Granularnost i selektivnost prometa	DA	NE	NE

Tablica 4.2 Rezultati testiranja konvergencije mreže

U skladu s očekivanjima, najbolje rezultate konvergencije izmjerili smo prilikom testiranja SDN topologije, a najlošije u topologiji tradicionalne mreže korištenjem STP protokola, gdje je vrijeme nedostupnosti mreže iznosilo 34 sekunde. U SDN mreži te u tradicionalnoj mreži s uključenim RSTP protokolom vrijeme nedostupnosti mreže bilo je nazamjetno manje (< 1 s) od testiranja u tradicionalnoj mreži korištenjem STP protokola, što pokazuje da se RSTP protokol može mjeriti po brzini konvergencije sa SDN rješenjima. Također, moguće je zamijetiti razlike u minimalnom, maksimalnom i prosječnom vremenu odgovora u sve tri kolone. To objašnjavamo boljim performansama OpenvSwitch virtualnog preklopnika unutar korištene virtualne okoline. U stvarnom okruženju rezultat latencije mreže u tradicionalnoj topologiji bio bi znatno povoljniji zbog upotrebe specifičnih ASIC čipova na samim hardverskim preklopnicima. Vrijeme konvergencije mreže te mogućnost granulacije prometa i dalje bi ostali isti zbog činjenice da se implementacija novih pravila prosljeđivanja preko upravljačke mreže može izvršiti simultano na svim SDN preklopnicima te da nije potreban rad sinkronizacijskih mehanizama kao kod RSTP protokola gdje se s povećanjem broja preklopnika u mreži povećava i vrijeme latencije kod konvergencije. Također, kod upravljanja prometa u tradicionalnoj mreži različite putanje za različite vrste prometa u mreži nisu moguće, u najboljem slučaju ciljamo sav promet u određenom VLAN-u. U SDN mreži možemo ciljati točno određenu vrstu prometa pa, kao u slučaju koji je testiran, sav promet s virtualne mašine CentOS1 možemo slati preko putanje S4-S2-S1, dok promet između drugih uređaja u mreži i dalje prolazi predefiniranom putanjom.

Prilikom dobivanja informacija o topologiji mreže također su zabilježeni rezultati koji idu u prilog korištenju SDN tehnologija. Vrijeme izmjereno prilikom dobivanja uvida u trenutačnu topologiju tradicionalne mreže korištenjem Solarwinds Network Mapper alata je 5 minuta 28 sekundi i 39 milisekundi, dok je vrijeme dobivanja uvida u trenutačnu topologiju SDN mreže u konzoli SDN kontrolera 5 sekundi i 14 milisekundi. Radi se o **98,47 %** boljem rezultatu u prilog SDN-a.

Ovime je pokazano da postoje prednosti u korištenju SDN tehnologija u okruženjima malih i srednje velikih organizacija, i to poglavito u vidu:

- **Centraliziranog upravljanja mrežnim prometom** – s jednog centralnog mjesta korištenjem adekvatnih aplikacija moguće je kontrolirati prometom u cjelokupnoj mreži, i to granularno, definiranjem zasebnih pravila za svaki tip prometa u mreži. Na isti način možemo postići funkcionalnost vatrozida bez korištenja dodatnih *middleware* uređaja za te svrhe
- **Brže konvergencije prometa u mreži** – prilikom konvergencije mreže u SDN topologiji zabilježeni su najbrži rezultati te najmanja latencija mreže. Iako razlike u zabilježenom vrijeme konvergencije SDN mreže i tradicionalne mreže kada se koristi RSTP protokol nisu jako izražene, rezultati pokazuju da se SDN može u svakom slučaju natjecati s tradicionalnom mrežom u ovom pogledu
- **Brže dobivanje stanja topologije u mreži** – dobivanja trenutačnih informacija o trenutačnoj topologiji u mreži jedna je od najvećih prednosti korištenja SDN-a, što je dokazano u testnom okruženju gdje je zabilježeno 98,47 % brže dobivanje stanja topologije mreže u SDN okruženju
- **Latencija mreže** – prilikom testiranja zabilježeni su bolji rezultati o stanju latencije mreže prilikom korištenja SDN topologije, no valja obratiti pozornost da se radi o virtualnom okruženju u kojem je korištenje softverskog preklopnika u vidu OpenvSwitcha očekivano dalo bolje rezultate od virtualiziranog IOS preklopnika.

4.5. Preporuke za daljnji rad i istraživanja

Za potrebe ovog rada napisana je SDN aplikacija u vidu *bash* skripte za proaktivno definiranje putanjama prometa u skladu s unaprijed definiranom klasifikacijom podataka u mreži. Aplikacija se može koristiti za planiranu rekonfiguraciju prometa u mreži u određenom razdoblju.

Daljnja istraživanja i rad valjalo bi usmjeriti prema razvoju aplikacije koja bi bila u stanju reaktivno plasirati *flow* zapise u tablice preklopnika ovisno o zauzetosti pojedinih resursa u SDN mreži. Openflow protokol nudi mogućnost prikupljanja statistike o sučeljima i *flow* zapisima u vidu brojača, a na taj način može se prikupljati broj procesuiranih paketa te broj poslanih bajtova na svakom sučelju preklopnika.

Predložena aplikacija prikupljala bi te podatke te u skladu s definiranim algoritmima mijenjala u realnom vremenu putanje u mreži, kada bi se zamijetila previsoka iskorištenost određenog linka ili *flow* zapisa u tablici.

Takva aplikacija trebala bi voditi računa o svim optimalnim putanjama u mreži kako bi se izbjegli slučajevi nastanka petlji ili prometa koji bi se slao na kriva sučelja.

U slučaju visoke zauzetosti resursa na pojedinim sučeljima izvršila bi rekonfiguraciju mreže koja bi slala dio prometa drugim putanjama u mreži.

Zaključak

U ovom radu obrađeni su osnovni koncepti i karakteristike softverski definiranih mreža uz pregled povijesti te ulogu softverski definiranih mreža danas i u budućnosti. Teoretski su obrađene osnovne razlike između tradicionalne i SDN mreže u vidu arhitekture, implementacije upravljačke i podatkovne ravni, implementacije i konfiguracije mreže, upravljanja i održavanja te troškova i koristi za organizacije.

SDN je nova paradigma u svijetu mreža koja mijenja način na koji su implementirane kontrolna i podatkovna ravan na mrežnim uređajima, kontrola nad mrežom konsolidirana je u vidu SDN kontrolera koji ima kompletnu kontrolu nad SDN mrežom koristeći Openflow protokol. Kontroler informacije o mreži daje na korištenje SDN aplikacijama na zasebnom sučelju korištenjem RESTful arhitekture, aplikacije mogu s mrežnim elementima komunicirati putem kontrolera, a tako mogu dobivati relevantna trenutačna stanja o mreži te ažurirati cjelokupnu mrežu s jedne centralne lokacije. SDN mreža omogućava prednosti centralizirane kontrole i uvida u stanje mreže na temelju kojeg se na lakši i troškovno povoljniji način može upravljati cjelokupnom mrežnom infrastrukturom.

Zbog razdvajanja kompleksnosti na različite uređaje te omogućavanje korištenja uobičajenog sklopovlja ostvaruje se povoljnija cijena operativnih i kapitalnih ulaganja.

U isto vrijeme zbog centralizirane kontrole moguće je lakše upravljati mrežom, kao i pratiti njezin rad, što omogućava da znatno manji broj administratora radi na implementaciji i upravljanju mrežom.

Velike IT korporacije kao što su Google i Facebook svoje su produkcijske mreže preselili na SDN okolinu, što im je omogućilo i do 60 % veću efikasnost u iskoristivosti svojih linkova. To se postiže centraliziranom kontrolom, separacijom slojeva, jednostavnijim uređajima na podatkovnom sloju, automatizacijom i virtualizacijom te korištenjem otvorenih standarda.

U praktičnom dijelu rada prikazan je rad SDN aplikacije napisane od strane autora kojoj je primarni cilj implementacija pravila upravljanja prometom u SDN mreži. Za potrebe demonstracije i usporedbe implementirane su SDN mreža i tradicionalna mreža bazirane na identičnoj troslojnoj mrežnoj topologiji. Prilikom rada aplikacije prikazana je autentifikacija na Aruba VAN SDN kontroler i implementacija pravila kontrole prometa u mreži. Za usporedbu upravljanja prometom u tradicionalnoj mreži korišteni su STP i RSTP protokoli.

Prilikom implementacije pravila te konvergencije mreže prikazano je da SDN tehnologijom možemo ostvariti bitno povoljnije vrijeme konvergencije mreže u usporedbi s tradicionalnom mrežom, pri čemu je za usmjeravanje prometom korišten standardni STP protokol. U usporedbi s RSTP protokolom, prikazano je identično vrijeme konvergencije, čime se dokazuje da upotrebom SDN tehnologija možemo implementirati sustav koji u svakom slučaju može konkurirati tradicionalnoj mreži u usporedbi vremena konvergencije mreže.

Prilikom rada aplikacije prikazana je mogućnost selektivnog upravljanja prometom, čime je dokazana granularnost prometa kakvu nije moguće ostvariti korištenjem tradicionalne mrežne topologije. Također, s obzirom na važnu ulogu koju prikaz trenutačne mrežne topologije u realnom vremenu može imati na odluke o mrežnom prometu, prikazano je da prilikom upravljanja prometom u SDN topologiji puno efikasnije i brže dolazimo do trenutačnog grafičkog prikaza mrežne topologije.

Popis kratica

SDN	*Software Defined Network*
STP	*Spanning tree protocol*
RSTP	*Rapid spanning tree protocol*
JSON	*JavaScript object notation*
BGP	*Border Gateway Protocol*
OSPF	*Open Shortest Path First*
EIGRP	*Enhanced Interior Gateway Routing Protocol*
CAM	*Content Addressable Memory*
REST	*Representional State Transfer*
API	*Application Programming Interface*
WAN	*Wide Area Network*
VRRP	*Virtual Router Redundancy Protocol*
L2	*Layer 2*
L3	*Layer 3*
L4	*Layer 4*
CDP	*Cisco Discovery Protocol*
LACP	*Link Aggregation Control Protocol*
HTTP	*Hypertext Transfer Protocol*
HTTPS	*Hypertext Transfer Protocol Secure*
URI	*Uniform Resource Identifier*
SNMP	*Simple Network Management Protocol*
MPLS	*Multiprotocol Label Switching*
BGP-LS	*BGP-Link State*
HP	*Hewlett-Packard*
NSA	*National Security Agency*
IBM	*International Business Machines*
SAL	*Service Abstraction Layer*
FIB	*Forwarding Information Base*
CPU	*Central Processing Unit*
VLAN	*Virtual Local Area Network*
TCP	*Transmission Control Protocol*
UDP	*User Datagram Protocol*

DSCP *Differentiated services*

ToS *Type of Service*

SSH *Secure Shell*

NVRAM *Non-Volatile Random Access Memory*

CLI *Command Line Interface*

GUI *Graphical User Interface*

LLDP *Link Layer Discover Protocol*

ICMP *Internet Control Message Protocol*

IT *Information Technology*

IP *Internet Protocol*

SMS *Short Message Service*

NMS *Network Management System*

CAPEX *Capital Expense*

OPEX *Operational Expense*

ONF *Open Network Foundation*

BASH *Bourne-again Shell*

MAC *Media Access Control*

FTP *File Transfer Protocol*

SCP *Secure copy*

LDAP *Lightweight Directory Access Protocol*

DPID *Datapath Identifier*

BPDU *Bridge Protocol Data Unit*

Popis slika

Popis tablica

Popis kôdova

Literatura

[1] K. Grey, T.D. Nadeau; SDN - Software Defined Networks; O'reilly; (2013); 978-1-4493-4320-2.

[2] C. Black, P. Goransson; Software Defined Networks; morgan kaufmann; (2014); 9780124166844.

[3] D. Bombai; Complete, practical SDN and Openflow fundamentals; Udemy; (2017); https://www.udemy.com/sdn-and-openflow-fundamentals/learn/v4/overview

[4] Brady, P.T. A statistical Analyis of On-off Patterns in 16 Conversation, *Bell System Technical Journal*, 47,1 (1998), 55-62.

[5] J. Cassey; Introduction to SDN and Openflow;INE; (2015); https://ine.com/collections/software-defined-networking/products/introduction-to-sdn-openflow

[6] D. Kreutz; F.M.W. Ramos; P. Verissimo; C.E. Rothenberg; S. Azodololky; S. Uhlik; Software Defined Networking - A Comprehensive Survey; IEEE; (2014);

[7] N. Freamster, J. Rexford, E. Zegura; The Road to SDN

[8] https://www.networkworld.com/article/3046457/data-center/the-networked-world.html, 24.07.2018

[9] X. Foukas, M.K. Marina, K. Kontovasilis; Software Defined Networking Concepts; (2015); str. 9

[10] https://www.wired.com/2012/04/going-with-the-flow-google/, 27.07.2018

[11] https://blogs.gartner.com/andrew-lerner/2017/08/29/the-state-of-sdn-september-2017/, 26.07.2018

[12] https://www.globalknowledge.com/us-en/content/articles/how-to-secure-cisco-routers-and-switches, 26.07.2018

[13] Software-Defined Networking using OpenFlow: Protocols, Applications and Architectural Design Choices, str. 303

[14] https://partnersolutions.skypeforbusiness.com/solutionscatalog/networking-hardware-infrastructure/hpe-network-optimizer, 30.07.2018

[15] http://h17007.www1.hpe.com/si/en/networking/solutions/technology/sdn/portfolio.aspx#.W1974tgzbOQ, 30.07.2018

[16] https://www.networkworld.com/article/2937787/sdn/nsa-uses-openflow-for-tracking-its-network.html, 31.07.2018

[17] https://thenewstack.io/sdn-series-part-iv-ryu-a-rich-featured-open-source-sdn-controller-supported-by-ntt-labs, 31.07.2018

[18] https://thenewstack.io/sdn-series-part-vi-opendaylight, 31.07.2018

[19] https://www.paessler.com/solutions, 04.08.2018

[20] https://www.solarwinds.com/network-configuration-manager, 04.08.2018

[21] https://telnetnetworks.wordpress.com/2014/10/30/sdn-and-network-management-infosim-stablenet-current-state-of-affairs-and-roadmap/, 04.08.2018

[22] https://www.investopedia.com/ask/answers/020915/what-difference-between-capex-and-opex.asp, 05.08.2018

[23] https://www.cisco.com/c/en/us/solutions/software-defined-networking/benefits.html, 06.08.2018

[24] http://docs.openvswitch.org/en/latest/topics/openflow/, 12.08.2018

[25] https://www.solarwinds.com/company/press-releases/solarwinds-network-topology-mapper-enhances-support-for-virtual-environments, 14.08.2018

[26] D Hanas, O Mohamed, Efficient Topology Discovery in Software Defined Networks: Revised, 2017

[27] https://www.restapitutorial.com/lessons/idempotency.html, 19.08.2018

[28] https://spring.io/understanding/REST, 19.08.2018

[29] https://stedolan.github.io/jq/, 20.08.2018

[30] http://blog.ine.com/2009/03/07/understanding-stp-convergence-part-i/, 29.08.2018

ALGEBRA
VISOKO
UČILIŠTE

UPRAVLJANJE PROMETOM U L2 SDN MREŽI

Pristupnik: Dario Posarić, 0321005402

Mentor: Silvio Papić, dipl. ing.