Automatizacija procesa konfiguracije preklopnika uporabom modela Zero-Touch Provisioning

Šimunić, Ivan

SVEUČILIŠTE U ZAGREBU
FAKULTET PROMETNIH ZNANOSTI

Ivan Šimunić

AUTOMATIZACIJA PROCESA KONFIGURACIJE PREKLOPNIKA UPORABOM MODELA ZERO-TOUCH PROVISIONING

DIPLOMSKI RAD

Zagreb, 2020.

SVEUČILIŠTE U ZAGREBU
FAKULTET PROMETNIH ZNANOSTI
POVJERENSTVO ZA DIPLOMSKI ISPIT

Zagreb, 7. svibnja 2020.

Zavod: **Zavod za informacijsko komunikacijski promet**
Predmet: **Planiranje telekomunikacijskih mreža**

DIPLOMSKI ZADATAK br. 6008

Pristupnik: **Ivan Šimunić (0135240895)**
Studij: Promet
Smjer: Informacijsko-komunikacijski promet

Zadatak: **Automatizacija procesa konfiguracije preklopnika uporabom modela Zero-Touch Provisioning**

Opis zadatka:

U diplomskom radu provest će se automatizacija procesa konfiguracije preklopnika uporabom modela Zero-Touch Provisioning (ZTP). Automatizacija preklopnika temeljit će se na preuzimanju datotečnih konfiguracija s poslužitelja. Pri tome će biti potrebno uspostaviti fizičko okruženje, odnosno laboratorij, u kojem se preklopnik koji podržava ZTP povezuje na lokalnu računalnu mrežu i locira Dynamic Host Configuration Protocol (DHCP) poslužitelj od kojeg, između ostaloga, dobiva IP adresu od Trivial File Transfer Protocol (TFTP) poslužitelja i datotečni put do skripte u programskom jeziku Python. Opisat će se značajke softverski definiranih mreža (SDN). Prikazat će se primjeri softverski definiranih pristupnih mreža i arhitektura softversko definiranog pristupa u žičnim i bežičnim mrežama. Analizirat će se ZTP modeli te opisati razvoj okruženja ZTP modela.

Mentor:

doc. dr. sc. Ivan Grgurević

Predsjednik povjerenstva za diplomski ispit:

Sveučilište u Zagrebu
Fakultet prometnih znanosti

DIPLOMSKI RAD

AUTOMATIZACIJA PROCESA KONFIGURACIJE PREKLOPNIKA UPORABOM MODELA ZERO-TOUCH PROVISIONING

AUTOMATION OF THE SWITCH CONFIGURATION PROCESS USING THE ZERO-TOUCH PROVISIONING MODEL

Mentor: doc. dr. sc. Ivan Grgurević

Student: Ivan Šimunić

JMBAG: 0135240895

Zagreb, rujan 2020.

AUTOMATIZACIJA PROCESA KONFIGURACIJE PREKLOPNIKA UPORABOM MODELA ZERO-TOUCH PROVISIONING

SAŽETAK

Tvrtke i organizacije koje danas provode digitalnu reformu postavljaju suvremenim mrežama nove i sve strože zahtjeve. Dodavanjem novih usluga i za njih potrebne mrežne infrastrukture, s ciljem zadovoljenja novih zahtjeva mreže, mrežnim se administratorima uz uobičajene svakodnevne zadatke upravljanja i optimizacije mreže znatno povećava opseg posla. Radi pojednostavljenja procesa implementacije novih mrežnih uređaja, unutar SD-*Access* koncepta razvija se rješenje automatizacije konfiguracije, *Zero Touch Provisioning* (ZTP). ZTP je moguće primijeniti i unutar tradicionalne arhitekture mreže što je i predmet ovog diplomskog rada. Inicijalni korak bila je izrada mrežnog okruženja primjenom alata *Emulated Virtual Environment Next Generation*. Kao eksperimentalno okruženje korištena je topologija mreže po uzoru na stvarna poslovna okruženja. U sklopu ovakve topologije definirani su potrebni poslužitelji i na različitim mrežnim lokacijama pridružena dva mrežna uređaja koji podržavaju ZTP model. Također, za provedbu procesa automatizacije bilo je potrebno izraditi skriptu u programskom jeziku *Python*. Pokretanjem predmetnih uređaja preuzima se skripta s postavljenog poslužitelja. Skripta se potom izvodi, a kao rezultat izvođenja primjenjuje se odgovarajuća konfiguracija na tim uređajima. Nakon provedbe ZTP modela i pregleda rezultata, potencijal koji omogućuje ovakvo rješenje u mrežama mnogo većih razmjera je evidentan. Primjena ovog modela zasigurno bi omogućila pojednostavljenje procesa instalacije novih uređaja na mrežu. Osim toga, model osigurava znatnu uštedu vremena potrebnog za instalaciju kao i financijskih sredstava, a naposljetku smanjuje i vjerojatnost pojave pogreške pri konfiguraciji. Rezultati ovog rada mogu pridonijeti boljem razumijevanju procesa implementacije ZTP modela s ciljem raširenije primjene istog.

KLJUČNE RIJEČI: Zero-Touch Provisioning (ZTP); Software-Defined Network (SDN); Software-Defined Access; Automatizacija; Konfiguracija mrežnih uređaja.

SUMMARY

Nowadays, companies and organizations that are implementing digital reform are placing new and increasingly stringent requirements on modern networks. By adding new services and the necessary network infrastructures in order to meet the new network requirements, network administrators, in addition to the usual daily tasks of network management and optimization, get significant increase of the scope of work. To simplify the process of implementing new network devices, a configuration automation solution Zero Touch Provisioning (ZTP) is being developed within the SD-Access concept. ZTP can also be applied within the traditional network architecture which is the main subject of this thesis. Initial step was creating a network environment using the Emulated Virtual Environment Next Generation tool. As an experimental environment, a network topology modelled on real business environments was used. Within this topology the required servers were defined and two devices which support the ZTP model were associated at different network locations. Also, to implement the automation process, it was necessary to create a script in the Python programming language. At startup, the two devices download the script from the defined server. The script is then executed, and as a result of this process the appropriate configuration is applied to these devices. After the implementation of the ZTP model and the review of the results, the potential provided by such solution in networks of much larger scales is evident. The application of this model would certainly simplify the process of installing new devices on the network. Besides, it would also provide significant savings in both installation time and financial resources, ultimately it would reduce the likelihood of configuration errors too. The results of this paper can contribute to the better understanding of the process of implementation of the ZTP model and therefore to its more widespread application in the future.

KEYWORDS: Zero-Touch Provisioning (ZTP); Software-Defined Network (SDN); Software-Defined Access; Automatization; Network Device Configuration.

Sadržaj

1. Uvod 1
2. Značajke softverski definiranih mreža 4
2.1. Usporedba softverski definiranih mreža s tradicionalnim mrežama 4
2.2. Arhitektura softverski definiranih mreža 6
2.2.1. Podatkovna razina 7
2.2.2. Kontrolna razina 9
2.2.3. Upravljačka razina 10
2.3. Primjena softverski definiranih mreža 11
3. Pregled softverski definiranog pristupa 13
3.1. SD-Access fabric 14
3.2. Pregled usluga i politike softverski definiranog pristupa 16
3.2.1. Virtualne mreže 17
3.2.2. Skalabilne grupe 17
3.2.3. Proširene podmreže 18
3.3. Automatizacija pristupnih mreža 20
4. Arhitektura softverski definiranog pristupa u žičanim i bežičnim mrežama 25
4.1. Elementi softverski definiranog pristupa 25
4.1.1. Čvorovi kontrolne razine 26
4.1.2. Granični čvorovi 27
4.1.3. Rubni čvorovi 27
4.1.4. Posredni čvorovi 28
4.1.5. Bežični LAN kontroler 28
4.1.6. Bežična pristupna točka 29
4.1.7. Identity Services Engine 29

4.1.8. DNA centar 30
4.2. Slojevita arhitektura softverski definiranog pristupa 30
4.2.1. SD-Access underlay 30
4.2.2. SD-Access overlay 31
4.3. Integracija bežične mreže sa softverski definiranim pristupom 31
4.3.1. SD-Access implementacija bežične mreže 32
4.3.2. Over-the-top implementacija bežične mreže 33
5. Pregled Zero Touch Provisioning modela 35
5.1. Postupak izvršavanja Zero Touch Provisioning procesa 35
5.2. Guest Shell okruženje 38
5.3. Python API 40
5.4. Python CLI modul 40
6. Razvoj okruženja Zero Touch Provisioning modela 42
6.1. Razvoj generalne topologije mreže 42
6.2. Konfiguracija mrežnih elemenata 44
6.3. Razvoj Python skripte 47
6.4. Proces izvršavanja ZTP modela 51
7. Zaključak 56
Popis kratica i akronima 58
Literatura 60
Popis slika 64

1. Uvod

Računalne mreže danas podržavaju značajno drugačije IT okruženje naspram unazad nekoliko godina. Broj mobilnih korisnika značajno raste, povećava se korištenje usluga temeljenih na „oblaku“ i dolazi do sve šire implementacije koncepta Interneta stvari. Tvrtke i krajnji korisnici očekuju da računalne mreže prate tehnološku evoluciju te podržavaju stalni rast mrežnih zahtjeva. Iako su mrežni stručnjaci uspijevali prilagoditi postojeće mreže novim izazovima neprestanom optimizacijom, implementacijom novih pristupa u dizajniranju mreže i uvođenjem novih značajki, računalne mreže se i dalje temelje na tradicionalnoj i nefleksibilnoj infrastrukturi.

Distribuirani kontrolni te transportni i mrežni protokoli unutar mrežnih uređaja predstavljaju ključne tehnologije koje omogućuju prijenos podataka unutar mreže. Za ostvarivanje željene razine funkcionalnosti mreže od mrežnih administratora se zahtjeva da konfiguriraju svaki uređaj zasebno putem sučelja naredbenog retka. Automatizacija konfiguracije i mehanizmi reakcije na događaje u mreži su praktički nepostojeći. Implementacija takvih mreža, iako možda ispunjava funkcionalne ciljeve, izrazito je kompleksna za razumijevanje, implementaciju, pronalazak problema i nadogradnju.

Kao pokušaj da se pojednostavi implementacija i upravljanje mrežom razvijen je koncept softverski definiranog pristupa (engl. *Software-Defined Access* ili *SD-Access*). Ideja SD-*Access* koncepta je da se omogući automatizacija *end-to-end* usluga u mreži, kao što su segmentacija klijenata, kvaliteta usluge i analitika. Nadalje, SD-*Access* pruža mogućnost automatizacije korisničke politike kako bi organizacije osigurale odgovarajuću kontrolu pristupa i korisničko iskustvo pri korištenju aplikacija za sve klijente na mreži. Također, SD-*Access* pruža i mogućnost automatizacije konfiguracije pri instalaciji novih uređaja na mrežu (engl. *Day-Zero Automation*).

Automatizacija konfiguracije ima potencijal značajno smanjiti vrijeme implementacije uređaja, ali i troškove koji se javljaju tijekom tog procesa. Istraživanja, poput onog od *Cisco*-a, sugeriraju da *Day-Zero Automation* može smanjiti troškove implementacije mreže i do 79 %.

Također, uslijed manje vjerojatnosti pojave pogreške pri konfiguraciji, moguće je smanjiti utrošak vremena u pronalasku problema za 50 %. Nadalje, 70 % svih pogrešaka u mreži i 35 %

ukupnog vremena za koje mreža nije bila funkcionalna posljedica su ljudske nesmotrenosti pri konfiguraciji što bi se moglo znatno smanjiti uvođenjem automatizacije implementacije mrežnih uređaja [1].

Model koji se koristi kao rješenje za automatizaciju konfiguracije novih uređaja na mreži naziva se *Zero-Touch Provisioning* (ZTP). Nekoliko autora je evaluiralo mogućnosti ZTP modela u raznim okruženjima i naglasilo prednosti istog, a o čemu će biti više u nastavku diplomskog rada. U istraživanju [2] autori navode prednosti ZTP modela pri automatizaciji konfiguracije, pri čemu autori kao posebnu prednost naglašavaju mogućnost konfiguracije velikog broja uređaja istovremeno. Tako je moguće inače distribuirani model mreže centralizirati upotrebom jedinstvene točke integracije, čime se otvaraju dodatne mogućnosti implementacije složenih mrežnih usluga u cijeloj domeni mreže.

Drugi autori također navode karakteristike ZTP modela i prednosti u okruženjima većih razmjera i pri upotrebi aplikacija temeljenih na „oblaku“. Autor u istom dokumentu navodi usmjerenost mrežnih proizvođača kao što su *Cisco* i *Jupiter* prema razvoju ovog koncepta te navodi implementacije ZTP modela od strane pojedinih mrežnih proizvođača. Također, navodi se mogućnost implementacije ovog modela s već postojećim mrežnim rješenjima i tehnologijama [3].

Pregledom literature utvrđeno je da postoje dokumenti koji su evaluirali ZTP model [2, 3], no ipak postoji nedostatak istraživanja koja se bave razvojem okruženja i primjenom tog istog modela. Stoga, svrha ovog istraživanja, osim prikaza funkcionalnosti, jest kroz proces implementacije ZTP modela u okruženje zasnovano na postojećim rješenjima predočiti sve prednosti navedenog čija bi primjena mrežnim stručnjacima u poslovnom okruženju značajno skratila vrijeme instalacije preklopnika, umanjila financijske troškove i vjerojatnost pojave ljudske pogreške.

Cilj istraživanja je provesti automatizaciju procesa konfiguracije preklopnika pri implementaciji u mrežu prema ZTP modelu. Automatizacija preklopnika temeljit će se na preuzimanju konfiguracijske skripte s odgovarajućih poslužitelja koja će se potom izvršiti na uređaju i uz pomoć konfiguracijske datoteke, također smještene na određenom poslužitelju, izvršiti konfiguraciju. Provedba ovog istraživanja zahtijevat će uspostavu laboratorija sa odgovarajućim mrežnim uređajima i poslužiteljima te razvoj skripte u programskom jeziku *Python.*

Struktura rada podijeljena je u sedam (7) poglavlja, uključujući uvod i zaključak. U drugom poglavlju predstavit će se općenito koncept softverski definiranih mreža i arhitektura na kojima se temelje. Također, prikazat će se usporedba tradicionalnih mreža sa softverski definiranim mrežama i prednosti nove paradigme mreže.

U trećem poglavlju opisat će se primjena koncepta softverski definiranih mreža u širokopojasnom pristupu i tehnologije koje se primjenjuju za ostvarenje ovog koncepta. Također, navest će se i opisat mogućnosti i usluge koje omogućuje softverski definiran pristup.

Četvrto poglavlje bavit će se prikazom slojevite arhitekture softverski definiranog pristupa. Također, u nastavku će se navesti i ukratko opisati elementi mreže koji se koriste u softverski definiranom pristupu.

U petom poglavlju slijedi pregled karakteristika ZTP modela. U ovom poglavlju opisat će se postupak izvršavanja ovog modela i okruženje koje je potrebno za izvršavanje.

Šesto poglavlje prikazuje razvoj okruženja za provedbu ZTP modela, što podrazumijeva razvoj mrežne topologije, prikaz konfiguracije uređaja, razvoj *Python* skripte i naposljetku provedbu ZTP modela te prikaz rezultata.

U Zaključku su sintetizirani svi dobiveni rezultati, dana zaključna razmatranja, planovi i smjernice za buduća istraživanja na temu automatizacije procesa konfiguracije preklopnika uporabom ZTP modela.

2. Značajke softverski definiranih mreža

Softverski definirane mreže (engl. *Software Defined Networks* - SDN) predstavljaju oblik mrežne arhitekture koja omogućuje fleksibilniji i jednostavniji pristup upravljanju mrežom i nastoji optimizirati mrežne resurse te prilagoditi mrežu zahtjevima korisnika. SDN razdvaja kontrolnu razinu od podatkovne i stvara programibilnu infrastrukturu čime se omogućuje pružanje novih tipova usluge kao što su segmentacija, virtualizacija i automatizacija.

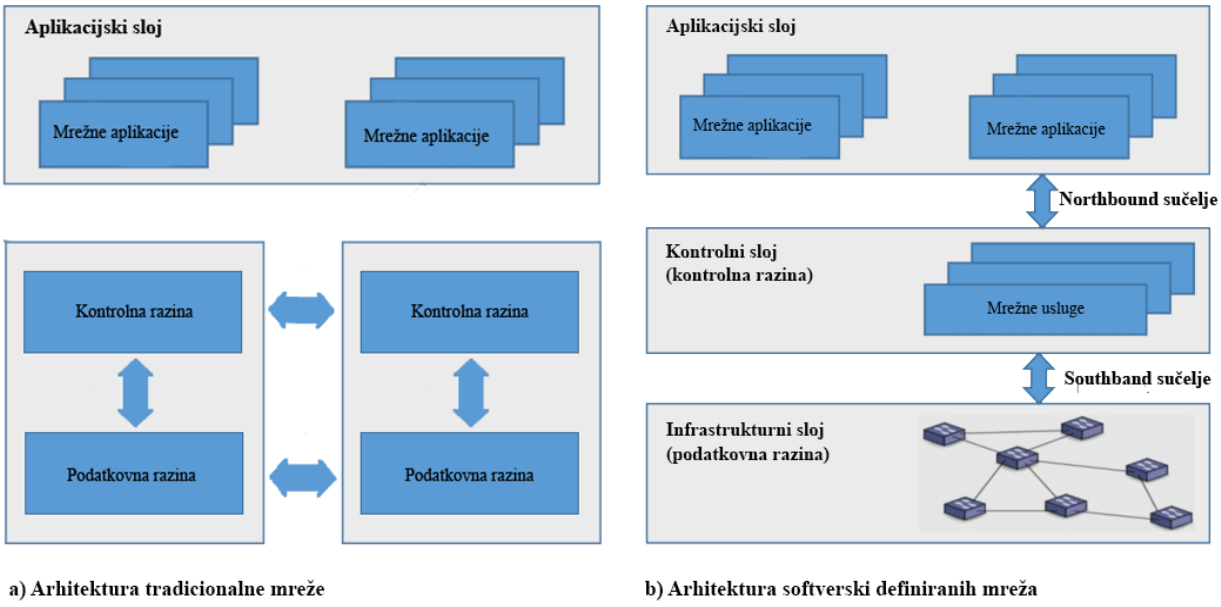
2.1. Usporedba softverski definiranih mreža s tradicionalnim mrežama

SDN je nova paradigma arhitekture mreže s centraliziranim pristupom koja zamjenjuje tradicionalne distribuirane mreže [4]. U tradicionalnim su mrežama kontrolni i transportni protokoli distribuirani na mrežnim uređajima te predstavljaju ključne elemente koji omogućuju protok prometa. Unatoč velikoj rasprostranjenosti, tradicionalne su IP mreže jako složene i zahtjevne za upravljanje. Upravljanje mreže u ovom slučaju podrazumijeva konfiguraciju individualnih mrežnih uređaja, što u okruženjima velikog broja uređaja i različitih proizvođača može postati jako složen proces. Automatizacija konfiguracije i odzivni mehanizmi praktički su nepostojeći zbog čega zadovoljavanje strožih zahtjeva mreže u dinamičnom okruženju predstavlja ogroman izazov [5].

Nadalje, arhitektura tradicionalnih mreža je vertikalno integrirana. Kontrolna razina (engl. *control plane*) koja odlučuje o upravljanju prometom i podatkovna razina (engl. *data plane*) koja prosljeđuje promet na osnovu odluka donesenih od strane kontrolne razine jesu zasebna odgovornost svakog mrežnog uređaja što smanjuje fleksibilnost i usporava razvoj mrežne infrastrukture. Tranzicija s IPv4 na IPv6[1] traje već više od desetljeća, a glavni razlog dugotrajnosti je upravo taj što razvijanje, evaluacija i implementacija novih mrežnih protokola u postojećim mrežama može trajati godinama zbog čega se njihovo uvođenje neprestano odgađa [5].

[1]IPv6 je novija verzija Internet Protokola koja postaje standardna verzija komunikacijskog protokola na Internetu. Trenutačno najrašireniji verzija je IPv4. Pojedine verzije Internet Protokola se razlikuju po načinu adresiranja, izgledu zaglavlja paketa, ali i brojnim drugim detaljima.

SDN donosi novi oblik mrežne arhitekture koji pruža priliku da se ukinu mnoga ograničenja tradicionalnih mreža. Prije svega, SDN ukida vertikalnu integraciju arhitekture mreže odvajanjem kontrolne razine od podatkovne razine, odnosno kontrolne logike od mrežnih uređaja koji prosljeđuju promet. Zatim, s odvajanjem kontrolne i podatkovne razine, mrežni uređaji postaju jednostavni uređaji sa svrhom prosljeđivanja prometa, a kontrolna logika se implementira na središnji kontroler. Na slici 1 prikazana je usporedba pojednostavljenih arhitektura tradicionalnih i softverski definiranih mreža [5].



Slika 1 Razlika između tradicionalne i SDN arhitekture. Izvor: [6]

Razdvajanje kontrolne i podatkovne razine moguće je postići programskim sučeljima između mrežnih uređaja i SDN kontrolera. Upotrebom aplikacijskog programskog sučelja (engl. *Application Programming Interface* – API), kontroler ima izravnu kontrolu nad elementima podatkovne razine kao što je prikazano na slici 1. Primjer jednog takvog sučelja je *OpenFlow*. Jedan *OpenFlow* preklopnik posjeduje jednu ili više tablica za upravljanje paketima. Na svaki podskup prometnog toka primijenit će se odgovarajuće pravilo nakon čega će se poduzeti određena akcija kao što je odbacivanje, prosljeđivanje, modificiranje i slično. Također, takav *OpenFlow*

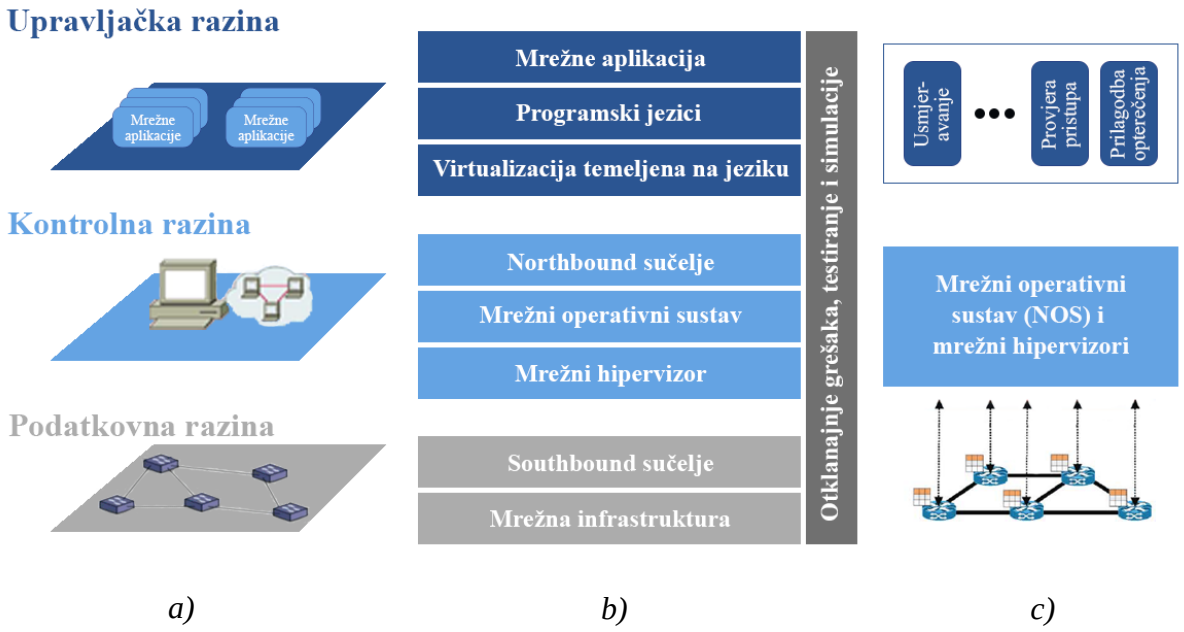
preklopnik prema naredbi kontrolera može se ponašati kao usmjernik, preklopnik, vatrozid ili može imati neku drugu ulogu [5].

Tradicionalna mreža, kao što je prikazano na slici 1, posjeduje veći broj nepovezanih kontrolera, odnosno mrežnih uređaja koji donose individualne upravljačke odluke, a ujedno i prosljeđuju promet, zbog čega je takva mreža sklonija neuspjehu. Ako su obje razine implementirane na jedan fizički uređaj, ova dva procesa su primorana dijeliti resurse tog uređaja čime dolazi do povećanja opterećenja mreže, procesora i memorije uređaja. Zbog toga razdvajanjem ovih procesa i implementiranjem zasebnih poslužitelja za upravljanje i nadzor uspostavlja se arhitektura mreže koja je dinamična, upravljiva, prilagodljiva i isplativa, što je čini poželjnom za današnje okruženje promjenjivih poslovnih zahtjeva [7].

2.2. Arhitektura softverski definiranih mreža

S obzirom na sve prethodno navedeno, softverski definirane mreže mogu se svesti na nekoliko osnovnih načela:

- Razdvajanje kontrolne i podatkovne razine – Razdvajanje procesa prosljeđivanja od procesa usmjeravanja, pri čemu mrežni uređaji postaju elementi koji samo prosljeđuju promet, a upravljačke odluke donosi središnji kontroler.
- Logički središnji kontroler – U usporedbi s distribuiranim kontrolerima u tradicionalnim mrežama, središnji kontroler posjeduje širu perspektivu mreže, stoga je u mogućnosti donositi bolje odluke.
- Izloženost apstraktnih mrežnih resursa i stanja vanjskim aplikacijama – Apstrakcija mrežnih resursa i stanja upotrebom aplikacija omogućuje programibilnost mreže. S informacijama o resursima i stanjima, aplikacije mogu definirati zahtjeve i zatražiti promjene na mrežnim uslugama preko mrežnog kontrolera.



Slika 2 Softverski definirane mreže prikazane kroz: a) razine, b) slojeve i c) arhitekturu dizajna sustava. Izvor: [5]

Arhitektura SDN-a identificira osnovne funkcionalnosti entiteta, operacije i informacije koje se razmjenjuju pomoću raznih sučelja. SDN arhitekturu čini kompozicija različitih slojeva, kao što je prikazano na slici 2b, pri čemu svaki sloj ima određenu funkciju. Iako se većina prikazanih slojeva nalazi uglavnom u svakoj implementaciji SDN-a, slojevi kao što su mrežni hipervizor[2] i virtualizacija temeljena na programskom jeziku implementirani su samo u posebnim slučajevima. Također, na slici 2 prikazan je i pregled prema razinama (2a) i arhitekturi dizajna sustava (2c). U nastavku će se opisati arhitektura SDN-a prema razinama.

2.2.1. Podatkovna razina

Podatkovna razina obuhvaća resurse koji se bave izravno klijentskim prometom, zajedno s resursima neophodnim za podržavanje virtualizacije, povezivosti, sigurnosti, dostupnosti i slično.

[2] Mrežni hipervizor predstavlja jedan od slojeva SDN arhitekture koji omogućuje višestrukim SDN kontrolerima upravljanje nad zasebnim dijelovima mreže.

Mrežna infrastruktura slična je onoj u tradicionalnim mrežama i čini je skup mrežnih uređaja (preklopnici, usmjernici i sl.). Glavna razlika u odnosu na tradicionalnu mrežu jest u tome što mrežni uređaji u ovom slučaju predstavljaju elemente koji prosljeđuju promet bez autonomnih upravljačkih odluka. Naime, mrežna inteligencija je sada odvojena od podatkovne razine i smještena je na središnji kontroler. Softverski definirane mreže temelje se na otvorenim sučeljima između razina, kao što je *OpenFlow*. Ovakav pristup omogućuje konfiguracijsku i komunikacijsku kompatibilnost između različitih uređaja. Tako otvorena sučelja omogućuju kontrolnim entitetima dinamično programiranje heterogenih uređaja što je u tradicionalnim mrežama, zbog različitih proizvođača i zatvorenih sučelja, iznimno težak zadatak [5].

Kontroleri i uređaji za prosljeđivanje prometa predstavljaju dva glavna elementa SDN arhitekture, pri čemu su uređaji podatkovne razine hardverski ili softverski elementi zaduženi za prosljeđivanje prometa, dok je kontroler složaj softvera koji radi na odgovarajućoj hardverskoj platformi. *OpenFlow* uređaji za prosljeđivanje zasnivaju se na tablicama koje se sastoje od pravila, akcija i brojača. Unutar jednog takvog uređaja, tok prometa kroz niz tablica definira način na koji će se upravljati prometom, a neke od akcija koje se poduzimaju su enkapsulacija i prosljeđivanje paketa prema kontroleru, odbacivanje paketa ili pak prosljeđivanje prema drugim tablicama [5].

Southbound sučelje je aplikacijsko programsko sučelje koje omogućuje komunikaciju između kontrolne i podatkovne razine. *OpenFlow* protokol je često prvi izbor za ovu namjenu. Upotreba protokola kao što je *OpenFlow* omogućuje interoperabilnost između opreme različitih proizvođača i osigurava tri vrste informacija za mrežni operativni sustav. Prije svega, osigurava informacije temeljene na događaju, odnosno kada se dogodi promjena na portu ili linku. Zatim, na *OpenFlow* uređajima se generira statistika mrežnog prometa koju prikuplja kontroler. Naposljetku, prosljeđivanje prometnog toka prema kontroleru u slučajevima kada uređaj ne zna što učiniti s dolazećim prometnim tokom [7].

Pored *OpenFlow* protokola, koriste se i drugi protokoli. Neki od češće korištenih su *Open vSwitch Database Protocol* (OVSDB), *Forwarding and Control Element Separation* (ForCES), *Protocol Oblivious Forwarding* (POF), *OpFlex control protocol*, *OpenFlow Config* (OF-Config), *OpenState*, *Revised OpenFlow Library* (ROFL), *Hardware Abstraction Layer* (HAL) te *Programmable Abstraction of Datapath* (PAD) [7].

2.2.2. Kontrolna razina

Implementacija središnjeg kontrolera s mrežnim operativnim sustavom (engl. *Network Operating System* - NOS) trebala bi riješiti probleme upravljanja mrežom i općenito smanjiti vjerojatnost pojave neočekivanih problema. NOS, kao i svaki operativni sustav, pruža apstrakciju, osnovne usluge i često korištene API-je za developere. Također, NOS omogućuje funkcionalnosti kao što su informiranje o topologiji i stanju mreže, zatim otkrivanje novih uređaja i distribucija mrežnih konfiguracija. Za definiranje mrežne politike administrator ne mora poznavati detalje nižih razina zbog čega takav sustav može znatno smanjiti kompleksnost kreiranja novih mrežnih protokola ili aplikacija i na taj način potaknuti inovacije i razvoj okruženja [8].

Središnji kontroler predstavlja ključni element u SDN arhitekturi koji omogućuje kontrolnim aplikacijama da generiraju mrežnu konfiguraciju na osnovu definirane mrežne politike. Mreža koja posjeduje jedan takav središnji kontroler posjeduje potencijalni rizik od padanja sustava usred prestanka rada takvog kontrolera. Također, problem takve mreže jest i skalabilnost. U tom slučaju mogu se implementirati distribuirani redundantni kontroleri. Distribuirani kontroleri mogu pridonijeti smanjenju rizika od pada mreže i omogućiti bolju skalabilnost [8].

Sučelja koja se koriste u slučaju distribuiranih kontrolera su *Eastbound* i *Westbound* sučelja. Svaki kontroler implementira vlastita sučelja. Funkcionalnost takvih sučelja uključuje prijenos podataka između kontrolera, algoritme za modele dosljednosti podataka i slično [5].

Northbound i *Southbound* sučelja predstavljaju ključne elemente apstrakcije u SDN ekosustavu. *Northbound* sučelje je mrežno apstrakcijsko sučelje za aplikacije i upravljačke sustave koji se nalaze na vrhu SDN složaja i pruža aplikacijama kontrolu nad mrežom. Ovo sučelje omogućuje osnovne mrežne funkcije kao što su računanje puta, izbjegavanje petlje, usmjeravanje i sigurnost. Mreže koje je moguće optimizirati preko ovog sučelja su upravitelji opterećenja, softverski definirane sigurnosne usluge i orkestracijske aplikacije [9]. Postoji više tipova *Northbound* sučelja, a uglavnom se koristi *Representational State Transfer* (REST) API [7].

Hipervizor je softver koji omogućuje kreiranje i pokretanje virtualnih strojeva te različitim virtualnim strojevima omogućuje da dijele hardverske resurse. Upotreba hipervizora omogućuje oblike poslovanja gdje klijenti raspoređuju resurse na zahtjev na dijeljenoj infrastrukturi, a s druge

strane omogućuje pružateljima usluga bolju iskoristivost fizičke infrastrukture. No, unatoč velikom broju prednosti koje nudi virtualizacija, u mrežama je slabo zastupljena pa se mreže konfiguriraju tako da se svaki uređaj posebno konfigurira [10].

Mrežna virtualizacija omogućuje virtualizaciju funkcija za mrežnu topologiju, IP adresni prostor i funkcije upravljanja. Mrežna virtualizacija omogućuje mrežnim administratorima da kreiraju i nadziru mrežne instance koje su logički odvojene od fizičke infrastrukture. Jedna od prednosti virtualizacijskih tehnologija je *Network Slicing.* Ova tehnologija omogućuje više logičkih mreža da dijele istu *OpenFlow* mrežnu infrastrukturu pa zbog toga osigurava apstrakcijski sloj koji olakšava rezanje podatkovne razine pri čemu bi se omogućilo postojanje više različitih mreža da istovremeno postoje [5].

2.2.3. Upravljačka razina

Upravljačka ili aplikativna razina sadrži jednu ili više aplikacija koje imaju kontrolu nad nizom resursa gdje pristup resursima omogućuju kontroler i različita sučelja. Aplikacije definiraju resurse i očekivano ponašanje mreže u kontekstu mrežne i poslovne politike. SDN je moguće implementirati u okruženjima različitih tradicionalnih mreža, od kućnih i poslovnih mreža sve do podatkovnih centara. Različita okruženja dovela su do pojave širokog spektra aplikacija. Iako postoji velik broj različitih aplikacija za karakteristična okruženja, većinu SDN aplikacija moguće je podijeliti u pet kategorija: prometno inženjerstvo, bežična mreža i mobilnost, mjerenja i nadzor, sigurnost i pouzdanost te umrežavanje podatkovnih centara [5].

Kao dio upravljačke razine pojavljuje se i sloj virtualizacije temeljene na programskom jeziku. Dvije osnovne karakteristike virtualizacijskih rješenja su omogućavanje modularnosti i omogućavanje različitih slojeva apstrakcije uz zajamčene opcije kao što je zaštita [8]. Virtualizacijske tehnike omogućuju različite poglede na fizičku infrastrukturu, tako se na primjer jedan virtualni preklopnik može sastojati od kombinacije više fizičkih. Zahvaljujući tome, developeri aplikacija ne moraju razmišljati o nizu preklopnika na kojima je potrebno primijeniti pravila prosljeđivanja. Umjesto toga će se mreža gledati kao jedan „veliki preklopnik“ [5]. Još jedan oblik virtualizacije temeljene na programskom jeziku je *Network Slicing*. Ovaj pojam

podrazumijeva dijeljenje mreže kompajlerom[3] na osnovu definicija aplikacijskog sloja. Izlaz kompajlera je monolitni kontrolni program koji posjeduje definicije za dijeljenje i konfiguracijske naredbe za mrežu te u tom slučaju nije potreban hipervizor za dinamično upravljanje dijelovima mreže. Statično rezanje se koristi pri implementacijama mreža sa specifičnim zahtjevima, obično tamo gdje su bolje performanse i jednostavna izolacija preferirana pred dinamičkim rezanjem [5].

2.3. Primjena softverski definiranih mreža

Jedna od glavnih prednosti koje SDN nudi je dinamički pristup informacijama o stanju mreže. Kao rezultat toga, aplikacijama je omogućeno definiranje usluga koje mreža treba podržati. Nadalje, SDN nudi prednosti kao što su mrežna virtualizacija, automatizacija konfiguracije i upravljanja te implementacija politika kao što je QoS[4] (engl. *Quality of Service*). Kombinacija visoko virtualiziranih okruženja unutar poslovnih mreža i porast mrežnog prometa negativno utječu na performanse tradicionalne mreže, čime se javlja potreba za upotrebom dinamičnih i skalabilnih mrežnih rješenja. SDN omogućuje organizacijama da znatno povećaju iskoristivost mreže i smanje troškove, posebno u *Bring Your Own Device*[5] (B.Y.O.D.) okruženjima te okruženjima aplikacija koje imaju velike zahtjeve propusnosti [11]. S obzirom na navedeno, tipična okruženja gdje se primjenjuje SDN su:

- Podatkovni centri
- Sveučilišni kampusi i
- Pružatelji usluga.

Podatkovni centri podržavaju promet i aplikacije s različitim zahtjevima za resurse, što može podrazumijevati zahtjeve za visoku propusnost ili pak određene sigurnosne zahtjeve. Mrežna virtualizacija predstavlja jednu od glavnih primjena SDN-a u podatkovnim centrima. Dva scenarija u kojima se mrežna virtualizacija koristi u podatkovnim centrima jesu za realizaciju više-

[3] Kompajler je računarski program koji čita program napisan u izvornom jeziku te ga prevodi u ciljani (najčešće mašinski) jezik.
[4] *Quality of Service* ili kvaliteta usluge odnosi se na mehanizam za kontrolu mrežnih resursa, odnosno sposobnost mreže u pružanju kvalitetnijih usluga.
[5] Podrazumijeva pravila tvrtke koja omogućuju zaposlenicima da ponesu vlastite mobilne terminalne uređaje do njihova radnog mjesta i koriste ih za pristup podacima tvrtke.

klijentskih mreža i proširenih mreža. SDN se primjenjuje za dinamično kreiranje odvojenih topološki ekvivalentnih mreža (više-klijentske mreže) u podatkovnom centru te kreiranje lokacijsko neovisnih mreža s omogućenom mobilnošću virtualnih strojeva i dinamičnom preraspodjelom resursa (proširene mreže) [11].

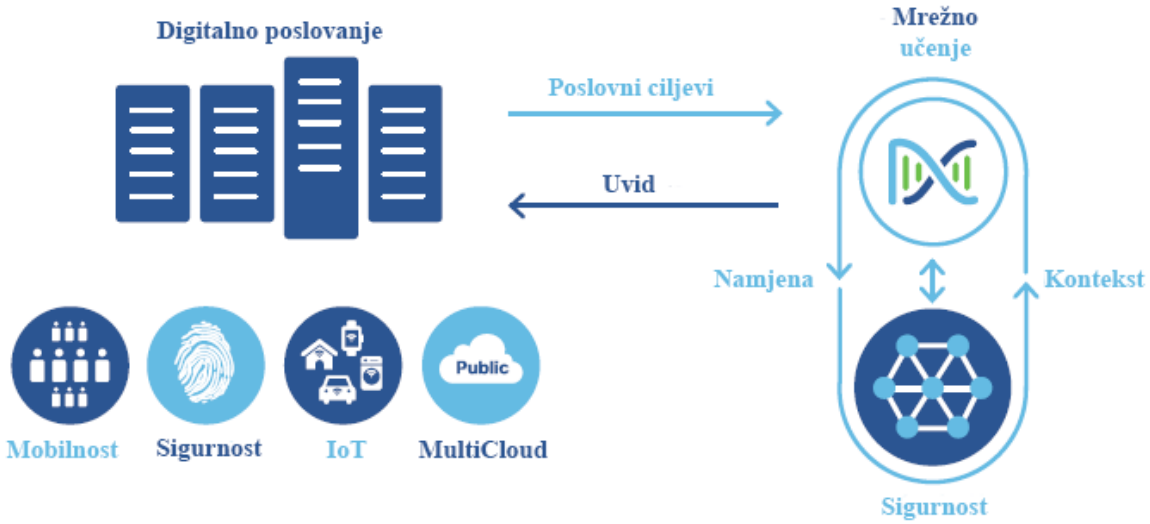
SDN na sveučilišnim kampusima omogućuje kreiranje logički izoliranih mreža upotrebom mrežne virtualizacije. Na sveučilištima javlja se karakteristična pojava velikog broja izvora mrežnog prometa, tako se razlikuju studenti, profesori, istraživačke grupe, administracija i slično te se virtualizacija primjenjuje za *Network Slicing* [11]. Uz navedeno, na sveučilišnim kampusima očekuje se generiranje velike količine prometa, pojava sve zahtjevnijih aplikacija glede kapaciteta prijenosa i velika gustoća heterogenih uređaja tome pridonose. Također, od mreže se očekuju funkcionalnosti u vidu mobilnosti i usmjeravanja temeljenog na aplikacijama. Upotrebom SDN-a sveučilišta, čije se mreže neprestano šire, mogu pružiti odgovarajuću kontrolu i upravljanje mreže te time udovoljiti rastućim korisničkim zahtjevima [12].

SDN pruža potpuno programibilno sučelje *operator-mreža* koje omogućuje operatoru da definira zahtjeve mreže bez konfiguracije bilo kojeg aspekta mreže niže razine. SDN posjeduje ovu fleksibilnost razdvajanjem kontrolne od podatkovne razine tako da distribucijski model kontrolne razine ne mora oponašati distribucijski sloj podatkovne razine. Neki od slučajeva uporabe su dinamično *Wide Area Network*[6] (WAN) preusmjeravanje, propusnost na zahtjev, virtualizacija mrežnih funkcija i zamjena postojećih *Customer Premises Equipment*[7] (CPE) s jednostavnijim verzijama te premještanje određenih funkcija i upravljanje kompleksnim prometom na podatkovne centre [11].

[6] WAN označava podatkovnu mrežu koja pokriva veće zemljopisno područje.
[7] CPE predstavlja terminalnu ili prateću opremu koja se nalazi na strani pretplatnika i povezana je sa mrežnim operatorom na demarkacijskoj točki.

3. Pregled softverski definiranog pristupa

Softverski definiran pristup (engl. *Software-Defined Access* – SD-*Access*) predstavlja novo rješenje za namjenski temeljeno umrežavanje (engl. *Intent-Based Networking* - IBS) koje proširuje koncept softverski definiranih mreža na širokopojasni pristup [13]. IBS tretira mrežu kao jedinstven sustav koji pruža translaciju i validaciju poslovnih ciljeva u mrežne zahtjeve i vraća izravan uvid kao što je prikazano slikom 3 [14].



Slika 3 Prikaz dijagrama namjenski temeljene mreže za digitalno poslovanje. Izvor: [14]

SDN, kao što je opisano u prethodnom poglavlju, predstavlja fizičko odvajanje kontrolne razine u mreži od podatkovne, gdje kontrolna razina vrši kontrolu mrežnih elemenata. SDN tako omogućuje stvaranje programibilne infrastrukture, a funkcije mrežne arhitekture, upravljanja, analitike i automatizacije postaju odgovornost središnjeg kontrolera. U širokopojasnom se pristupu, također, javlja potreba za migracijom kontrolnih i upravljačkih funkcija s *firmaware*-a[8] u namjenskoj mrežnoj opremi na softverske kontrolere koji su smješteni na *cloudu* [13]. *SD-Access*

[8] Označava softver koji je ugrađen u hardverski uređaj. Obično se nalazi u *flash* ROM-u ili u obliku binarnog podatka koji se može postaviti na postojeći hardver.

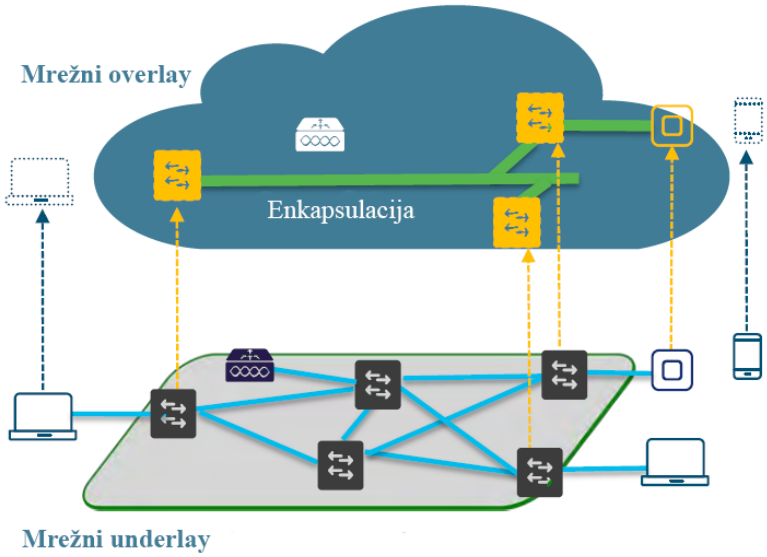
predstavlja rješenje za programibilnu mrežnu arhitekturu koja pruža softverski temeljenu politiku i segmentaciju od ruba mreže pa do aplikacije.

Ovo rješenje omogućuje definiranje dizajna, politike, automatizaciju implementacije mrežnih elemenata te analitičke izvještaje za inteligentnu žičanu i bežičnu mrežu. U organizacijama se mreža može prostirati na više domena i lokacija, kao što su kampusi i podružnice, svaka s karakterističnim uređajima, uslugama i politikama. SD-*Access* pruža *end-to-end* arhitekturu koja omogućuje dosljednost u smislu povezivosti, segmentacije i politike na različitim lokacijama. Prethodno navedeno moguće je realizirati upotrebom *network fabric* topologije [14].

SD-*Access* koncept može se primijeniti na bilo koju vrstu širokopojasnog pristupa: *Digital Subscriber Line* (DSL), kabelski modemi, *Fiber to the Premises* (FTTP) ili druge pristupne tehnologije.

3.1. SD-Access fabric

Network fabric je pojam koji se upotrebljava za topologiju koja pruža žičanim i bežičnim mrežama programibilni *overlay*, omogućuje mrežnu virtualizaciju i pruža mogućnost fizičkom dijelu mreže da ugosti jednu ili više logičkih mreža. *Overlay* pruža logičku topologiju koja se implementira povrh fizičkog *underlay* sloja [15]. Zajedno logički i fizički dio mreže omogućuju pružanje različite funkcionalnosti prosljeđivanja prometa, segmentacije, dostupnosti i slično. Dakle, *fabric* predstavlja kombinaciju *overlay* i *underlay* slojeva mreže (slika 4). *Underlay* je sloj dodijeljen fizičkim uređajima za prosljeđivanje prometa, dok s druge strane *overlay* predstavlja virtualni sloj gdje su žičani i bežični klijenti logički međusobno povezani te na kojem se pružaju usluge i primjenjuju politike [1].



Slika 4 Prikaz overlay i underlay slojeva. Izvor: [14]

Od mreže se uglavnom očekuje da bude stabilna, predvidljiva i uvijek u funkciji, no s druge strane očekuje se i stalno uvođenje novih mrežnih usluga i funkcionalnosti kako bi se zadovoljili uvijek rastući poslovni zahtjevi. Kada je u pitanju mreža koju čini samo fizički sloj, svaka promjena na mreži iziskuje konfiguraciju fizičkih uređaja zbog čega svaka promjena može narušiti funkcionalnost mreže čime se ograničava uvođenje novih usluga i samim tim razvoj mreže.

Fabric mreže razdvajaju funkcije prosljeđivanja prometa od upravljanja istim. U ovom slučaju, na *underlay* sloju nalaze se mrežni elementi, preklopnici i usmjernici, zaduženi za prosljeđivanje prometnih tokova. Povezanost mrežnih elemenata postiže se upotrebom IP konekcije i protokola usmjeravanja čime se pruža stabilnost i poboljšavaju performanse mreže dok se smanjuje vrijeme konvergencije [15]. *Underlay* mreža se nakon implementacije rijetko konfigurira, osim u slučaju dodavanja uređaja ili linkova što u *fabric* mrežama pruža pouzdanu, stabilnu i jednostavnu osnovu za logički *overlay* [1].

Overlay je logička topologija koja je smještena povrh fizičkog sloja. *Overlay* omogućuje virtualizirane usluge kao što su segmentacija i pruža mogućnost mobilnosti unutar *fabric* topologije. *Overlay* dodatno omogućuje pružanje jednakih usluga za žičane i bežične klijente, kao što su virtualizacija, sigurnosne značajke i segmentacija, uz jednake mogućnosti i performanse [1]. Također, upotrebom novog oblika enkapsulacijske tehnologije, kao što je *Virtual Extensible* LAN

(VXLAN) pruža se mogućnost implementacije virtualnih mreža (engl. *Virtual Networks* - VN) i oznaka skalabilnih grupa (engl. *Scalable Group Tag* - SGT) za uspostavljanje makrosegmentacije ili mikrosegmentacije unutar mreže [14].

Jedna od glavnih prednosti upotrebe *fabric* topologije je u tome što pojednostavljuje upravljanje mrežom. U tradicionalnim mrežama primjena sigurnosne politike zasnovana je na upotrebi pristupnih kontrolnih lista (engl. *Access Control List* - ACL) i virtualnih LAN-ova (VLAN). Klijenti koji se povezuju na mrežu pridružuju se odgovarajućem VLAN-u dok su se upotrebom ACL-ova na mrežnim uređajima ili upotrebom vatrozida primjenjivale sigurnosne politike. Negativna posljedica upotrebe takvih metoda za implementiranje sigurnosne politike jest to što se u poslovnim okruženjima može javiti jako velik broj VLAN-ova i linija naredbi pri definiranju ACL-ova, što može otežati implementaciju i održavanje [1].

U *fabric* mrežama koristi se drugačiji pristup od onoga u tradicionalnim pristupnim mrežama. Svi klijenti koji se povezuju na *fabric* mrežu autentificiraju se i na osnovu toga se pridružuju određenoj skalabilnoj grupi, dodjeljuje im se uloga i mapiraju se na određenu virtualnu mrežu. Kada se klijentski paketi enkapsuliraju u VXLAN zaglavlje dodaju se VN i SGT oznake te se implementacija mrežne politike zasniva na tim informacijama [1]. U SD-*Access* mrežama IP adrese koriste se samo za provjeru dostupnosti dok se lokacija utvrđuje uporabom protokola *Locator* ID *Separation Protocol* (LISP), koji koristi klijentske oznake ili *Endpoint Identifiers* (EID) i oznake uređaja ili *Routing Locators* (RLOC) [16].

3.2. Pregled usluga i politike softverski definiranog pristupa

Prilikom definiranja usluga i politike mreže bitno je započeti s poslovnim pokretačima koji kreiraju zahtjeve. Nekada su jedini mrežni zahtjevi bili osiguravanje visoke dostupnosti i zadovoljavajuće brzine pristupa. S pojavom novih trendova u računalstvu i umrežavanju, usluge i politike su također evoluirale. SD-*Access* sigurnosna politika se pojavljuje u dva tipa: politika virtualnih mreža i politika temeljena na grupama.

Jedna od osnovnih sigurnosnih usluga koje SD-*Access* nudi jest segmentacija određenih grupa klijenata od drugih. Mrežnu segmentaciju omogućuje VXLAN enkapsulacija upotrebom

određenih polja u VXLAN zaglavlju: VN i SGT oznaka. SD-*Access* pruža mogućnost implementacije hijerarhijske segmentacije upotrebom makrosegmentacije i mikrosegmentacije [14].

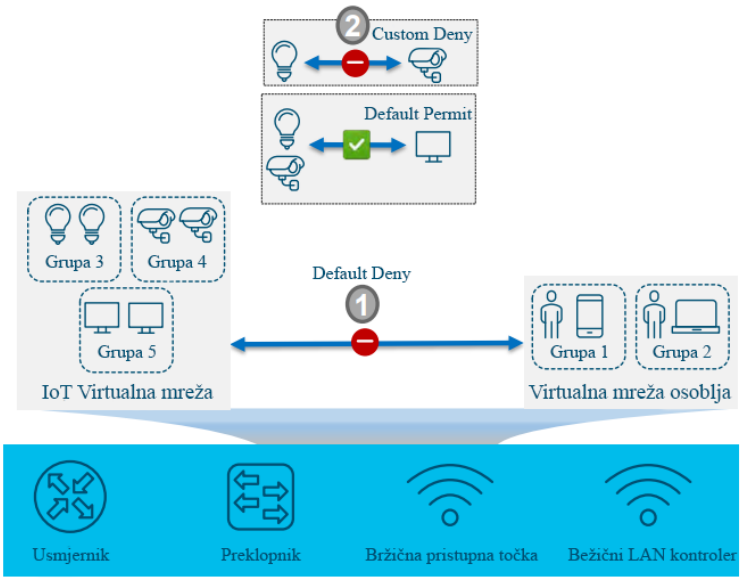
3.2.1. Virtualne mreže

VN koncept u SD-*Access* mrežama omogućuje makrosegmentaciju. Pod pojmom makrosegmentacije podrazumijeva se potpuna izolacija cijele grupe klijenata i njihovog prometa unutar jednog VN-a od drugih VN-ova u istoj mreži. Klijenti se smještaju u VN-ove na osnovu njihovog identiteta, a informacija koja identificira VN klijenta u SD-*Access* mreži smještena je u VXLAN zaglavlju. Različiti VN-ovi ne zahtijevaju zasebne tablice usmjeravanja, budući da se LISP koristi za pružanje informacija o usmjeravanju. Politika mreže u slučaju VN-ova implementira se na osnovu modela da je sav promet između VN-ova blokiran, osim ako se ne definira drugačije. Na osnovu toga može se reći da VN predstavlja prvu razinu segmentacije koja osigurava da je komunikacija između različitih VN-ova onemogućena [17].

3.2.2. Skalabilne grupe

Upotreba skalabilnih grupa unutar SD-*Access* mreže pruža dalje mogućnost mikrosegementacije, odnosno mogućnost filtriranja i kontrole prometa unutar VN-ova. Skalabilne grupe koriste oznaku od 16 bitova, odnosno SGT, koja je smještena u VXLAN zaglavlju i koja identificira grupu kojoj klijent pripada. Za kontrolu pristupa određenim resursima, a drugim klijentima unutar skalabilne grupe na osnovu identiteta i uloge koriste se pristupne liste skalabilnih grupa [17].

Skalabilne grupe koriste model komunikacije između grupa u kojem je svaka komunikacija dopuštena, osim ako se ne definira drugačije, što je suprotno od onoga što se implementira kod VN-ova. Primjer upotrebe skalabilnih grupa unutar virtualnih mreža predočen je slikom 5.



Slika 5 Virtualne mreže i skalabilne grupe u SD-Access mreži. Izvor: [18]

Pružanjem dviju razine segmentacije, SD-*Access* omogućuje značajno sigurniju implementaciju mreže u poslovnim okruženjima, a s druge strane omogućuje jednostavan pristup dizajniranju, implementaciji i održavanju. Također, uporabom ovog koncepta mrežna politika u SD-*Access* okruženju vezana je uz identitet klijenta, a ne IP adresu što olakšava primjenu mrežne politike pri *roamingu*.

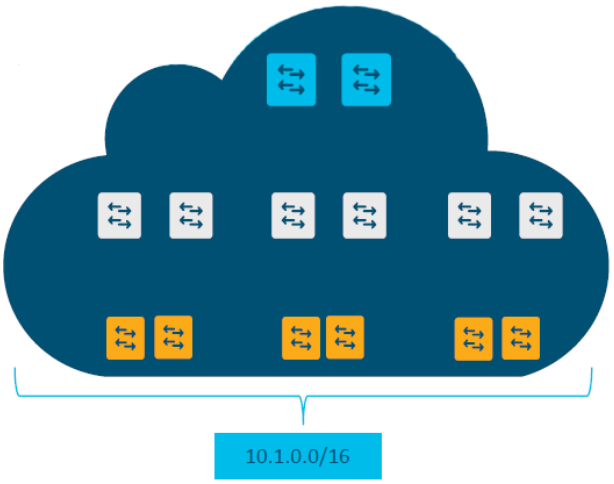
3.2.3. Proširene podmreže

Mnoga poslovna okruženja imaju potrebu za pružanjem jedinstvene podmreže koja će se prostirat na većem području kampusa, a razlozi mogu biti razni. Implementacija takve podmreže u tradicionalnim mrežama zahtjeva proširenje VLAN-ova na više pristupnih mrežnih uređaja što iziskuje njihovo povezivanje u jedinstvenu domenu drugog sloja *Open Systems Interconnection*[9] (OSI) modela. Tako proširena domena drugog sloja uz postavljanje redundantnih linkova stvara

[9] OSI referentni model je apstraktni, slojeviti model koji služi kao preporuka stručnjacima za razvoj računalnih mreža i protokola. OSI model je podijeljen u sedam slojeva, gdje svaki sloj opisuje skup povezanih funkcija koje omogućuju jedan dio računalne komunikacije. Svih sedam slojeva zajedno, prikazuju tok podataka od izvora prema odredištu.

petlje zbog čega je potrebno implementirati neku od inačica *Spanning Tree Protocol*-a (STP) koji onemogućuje pojavu petlje blokiranjem određenih portova. Iako je ovakav način implementacije funkcionalan, nosi određene rizike pa predstavlja lošu iskoristivost mrežnog kapaciteta zbog velikog broja blokiranih portova. Također, upotrebom jedinstvene domene drugog javlja se opasnost da određeni kvar utječe na cjelovitu mrežu umjesto na samo jedan njen dio. Zbog svega navedenog, implementacija ovog rješenja često se izbjegava [1]. S druge strane, SD-*Access* omogućuje implementaciju proširenih podmreža na jednostavan način bez upotrebe STP protokola.

Kao što je prikazano na slici 6, u SD-*Access* okruženju jedinstvena podmreža je implementirana na sve pristupne uređaje na čitavoj mreži. Na ovaj način, klijenti koji se povežu na bilo koji pristupni mrežni uređaj na osnovi identiteta i uloge pridružuju se odgovarajućoj podmreži, pri čemu će klijent posjedovati istu IP i *Media Access Control* (MAC) adresu *default gateway*[10] na bilo kojoj pristupnoj lokaciji. Ovo se ostvaruje upotrebom *Distributed Anycast Default Gateway* funkcionalnosti koju omogućuje SD-*Access*. Ovaj koncept dodatno olakšava mobilnost klijenata i pojednostavljuje implementaciju te upravljanje mrežom [1].



Slika 6 Prikaz primjera proširene podmreže. Izvor: [18]

[10] Adresa predefiniranog izlaza za određenu podmrežu.

Iako se STP protokol i dalje primjenjuje, domena drugog sloja OSI modela nije proširena na ostatak mreže, a *broadcast* domena ograničena je samo na uređaje unutar jednog mrežnog ormara. Nadalje, upotrebom *Distributed Anycast Default Gateway* koncepta, nema potrebe za implementacijom redundantnog *default gatewaya* kao što je bilo potrebno u tradicionalnim mrežama. Zbog toga nema potrebe za upotrebom protokola, kao što su *Hot Standby Router Protocol* (HSRP) i *Virtual Router Redundancy Protocol* (VRRP) u tu svrhu, čime se dodatno smanjuje potreba za redundantnim implementacijama i smanjuje se kompleksnost mreže te njenog upravljanja [1].

Budući da se SD-*Access* temelji na fizičkoj topologiji u kojoj su mrežni uređaji povezani linkovima trećeg sloja OSI modela i implementirani protokoli usmjeravanja. Kao posljedica toga moguće je primijeniti *Equal-Cost Multi-Path Routing* (ECMP), odnosno strategiju usmjeravanja gdje se paketi prosljeđuju preko više najbolje ocjenjenih ruta. Primjenom ove strategije pruža se mogućnost iskorištavanja svih linkova, a da se pritom mrežni promet jednako prosljeđuje preko više linkova. Upotrebom *overlay*-a i linkova trećeg sloja značajno se poboljšavaju performanse mreže, osigurava veća stabilnost, konvergencija i predvidljivost mreže.

3.3. Automatizacija pristupnih mreža

SD-*Access* automatizacija pruža mogućnost definiranja i upravljanja SD-*Access* politike na razini grupe te automatizaciju konfiguracije politike. Automatizacija generalno može biti definirana kao tehnologija ili sustav koji provodi akcije bez ljudske pomoći. Automatizacija oslobađa mrežne administratore od izvođenja vremenski zahtjevnih i repetitivnih konfiguracijskih zadataka te pruža standardiziranu i bez grešaka konfiguriranu *underlay* mrežu. Također, automatizacija ubrzava proces izgradnje *overlay* mreže bez potrebe za planiranjem i implementacijom tradicionalne mreže [19].

Automatizacija pruža sljedeće prednosti [19]:

- *Zero-Touch Provisioning* – Mrežni uređaji dinamično se otkrivaju, pridružuju i automatiziraju iz svog tvorničkog stanja u potpunu integriranu mrežu.
- *End-to-end* topologija – Dinamično otkrivanje novih mrežnih sustava i njihove fizičke konekcije mogu biti modelirane te programirane. Takvi novi sustavi mogu biti automatizirani adresiranjem na trećem sloju OSI modela uz upotrebu protokola usmjeravanja kako bi dinamično izradili *end-to-end* topologiju usmjeravanja.
- Elastičnost – Omogućuje se redundancija na razini sustava i upotreba automatizacije najbolje prakse za postizanje elastičnosti mreže tijekom planiranih i neplaniranih događaja u mreži.
- Sigurnost – Parametri zaštite mrežnog pristupa i infrastrukture su automatizirani čime se pruža odgovarajuća sigurnost pri inicijalnoj implementaciji.
- Usklađenost – Automatizacija mreže pomaže u eliminaciji ljudskih grešaka, kao što su pogreške u konfiguraciji i nedosljednost pravila te postavki koje iscrpljuju resurse sustava. Automatizacija pruža usklađenost mrežne infrastrukture automatizacijom globalno upravljivih parametara.

Proces implementacije automatizacije u LAN mrežama zahtjeva pristup po koracima [19]:

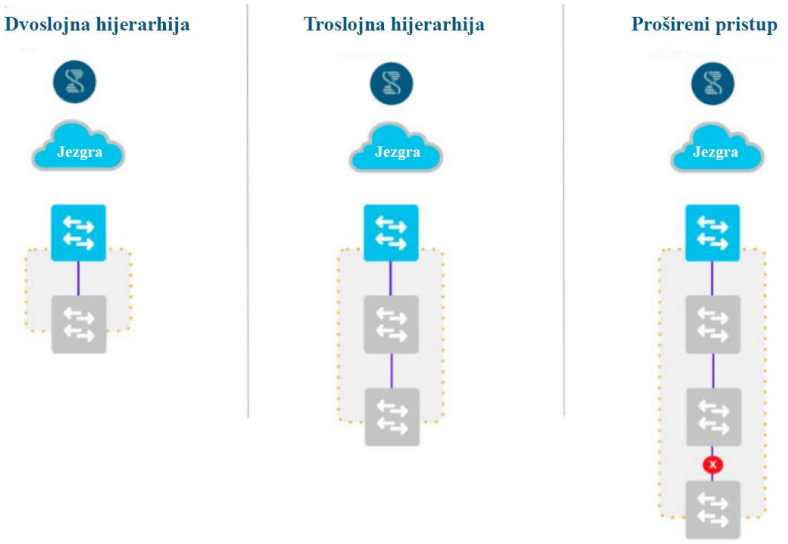
- Planiranje – Ovaj se korak odnosi na razumijevanje različitih uloga u domeni automatizacije pristupnih mreža, uslijed čega se planira okruženje i IP domena te se određuju preduvjeti za *seed* uređaje.
- Dizajn – Podrazumijeva dizajniranje i izradu okruženja, konfiguraciju globalnih mrežnih usluga i usluga na razini okruženja. Potom se konfiguriraju akreditive globalnih uređaja i naposljetku dizajniranje globalnih domena IP adresa i dodjeljivanje opsega adresa automatizacijskom procesu pristupne mreže.
- Otkrivanje – Uključuje otkrivanje *seed* mrežnih uređaja.
- Opskrba konfiguracijom – U ovom koraku započinje proces automatizacije prilikom čega se prosljeđuje privremena konfiguracija na *seed* uređaje, otkrivaju ostali mrežni uređaji na kojim se nadograđuje *system image* i na koje se također prosljeđuje inicijalna konfiguracija. Korak opskrbe završava pretvaranjem *point-to-point* linkova u linkove trećeg sloja OSI modela.

Kreiranje plana automatizacije predstavlja prvi korak u kreiranju odgovarajućeg *underlay* sloja mreže. Pri kreiranju odgovarajućeg plana potrebno je prije svega poznavati određene uloge u procesu automatizacije. U nastavku će biti ukratko opisan postupak implementacije automatizacije.

Na samom početku potrebno je implementirati *seed* uređaje koji predstavljaju početnu točku preko koje se otkrivaju uređaji na nižim razinama i pridružuju mreži. *Seed* uređaji mogu biti automatizirani upotrebom tehnologija za inicijalnu konfiguraciju kao što su *Cisco Plug-n-Play* (PnP) i *Zero-Touch Provisioning* (ZTP) ili se mogu ručno postaviti. Dovoljan je jedan *seed* uređaj, ali može se koristiti i više njih [19].

Dizajn LAN mreže varira i uvelike ovisi o veličini mreže, fizičkim linkovima i slično. Uglavnom se preporučuje koristiti hijerarhijski dizajn u poslovnim okruženjima kako bi se pružila skalabilnost i redundancija na svakom sloju mreže[11]. Kao dio inicijalnog planiranja potrebno je odrediti fizičku topologiju. Platforme kao što su *Cisco* DNA (engl. *Digital Network Architecture*) prilikom automatizacije novih uređaja podržavaju najviše do dva skoka u mreži od inicijalnog graničnog uređaja. Drugim riječima, prilikom kreiranja *underlay* sloja biti će potrebno postaviti granični *seed* uređaj na jezgrenom ili distribucijskom sloju, budući da uređaji koji se nalaze dalje od dva skoka mogu biti otkriveni, ali ne i automatizirani (slika 6). Također, automatizacija zahvaća samo izravno povezane susjede. Na primjer, kada se dodaju uređaji pristupnog i distribucijskog sloja, koji su povezani na *seed* uređaje jezgrenog sloja, izvršit će se automatizacija svih uređaja. Za razliku od ovog primjera, u slučaju da su uređaji distribucijskog sloja već automatizirani, uređaje jezgrenog sloja nije moguće koristiti za automatizaciju uređaja pristupnog sloja. U tom slučaju uređaji distribucijskog sloja ponašaju se kao *seed* uređaji i vrše automatizaciju [19].

[11] Uglavnom postoje tri sloja mreže, a to su: pristupni, distribucijski i jezgreni slojevi. U mrežama kao što su one na kampusu gdje postoji više objekata koristi se dizajn s tri sloja, dok se u nekim manjim okruženjima koristi dizajn s dva sloja, bez jezgrenog sloja (engl. *Collapsed Core*).



Slika 7 Prikaz granice automatizacije koju podržava Cisco DNA Centar. Izvor: [19]

Nakon što završi proces automatizacije i dodavanja novih uređaja u mrežu, započinje proces konfiguracije linkova trećeg sloja OSI modela. Proces započinje konverzijom linkova drugog sloja u linkove trećeg sloja OSI modela prema mrežnom grafu koji je kreiran pri pridruživanju uređaja na mrežu. Zatim, link prema uređaju višeg sloja zaprima IP adresu i na usmjerniku se konfigurira određeni usmjerivački protokol [19].

Također, potrebno je na aplikaciji središnjeg kontrolera kreirati odgovarajući model objekata, etaža i okruženja. Pod time se podrazumijeva definiranje načina na koji će primarni i pridruženi *seed* uređaj biti povezani s novim uređajima, kao na primjer da li će biti na istoj lokaciji ili će pratiti hijerarhijski model. Nadalje, potrebno je odlučiti kako će skup IP adresa biti dijeljen između različitih lokacija, objekata i etaža, pri čemu je moguće koristiti jedan skup za različita okruženja, ili koristiti različite skupove IP adresa.

Kreiranje skupova IP adresa za automatizaciju pristupnih mreža započinje kreiranjem globalnog skupa adresa, nakon čega slijedi definiranje skupa adresa za određenu lokaciju pristupne mreže. Za automatizaciju upotrebljava se lokacijski temeljen skup adresa. Taj skup se koristi za sljedeće alokacije:

- Dio skupa adresa rezervira se za privremeni *Dynamic Host Configuration Protocol* (DHCP) poslužitelj. Veličina skupa određenog za DHCP server ovisi o veličini inicijalnog

skupa. Ovaj skup rezervira se samo za vrijeme trajanja procesa otkrivanja uređaja, a kada se proces privede kraju rezervirani se skup adresa oslobađa i predaje primarnom skupu.

- Drugi dio skupa IP adresa koristi se u konfiguraciji linkova između povezanih uređaja koji sudjeluju u procesu otkrivanja uređaja, a to su *seed* uređaji, pridruženi *seed* uređaji i otkriveni uređaji. Linkovi između tih uređaja su konfigurirani za omogućavanje protokola usmjeravanja.
- Posljednji dio skupa adresa alocira se za *loopback*[12] adrese otkrivenih uređaja. Također, adrese iz skupa se pridružuju i *seed* uređajima u slučaju da ne posjeduju *loopback* adresu.

Potrebno je naglasiti da se jedan skup adresa može iskoristiti za višestruke procese automatizacije. Također, potrebno je izbjegavati upotrebu skupa adresa koji se već koristi negdje u mreži.

Prije pokretanja procesa automatizacije potrebno je definirati i odgovarajuće konfiguracije *Simple Network Management Protocol* (SNMP) protokola i korisničkog sučelja za različita okruženja, a ona se obavlja na središnjem kontroleru. SNMP predstavlja protokol aplikacijskog sloja OSI modela koji omogućuje mrežnim uređajima razmjenjivanje informacija međusobno, ali i s uređajima izvan mreže, na osnovu kojeg administratori upravljaju mrežom, pronalaze probleme te ih rješavaju [20].

Seed uređaji dakle predstavljaju točku s koje započinje proces automatizacije i s koje se obavlja otkrivanje novih uređaja. Kao što je već spomenuto, na *seed* uređaj u mogućnosti je implementirati konfiguraciju određenim automatizacijskim procesima ili ručno. Konfiguracija *seed* uređaja podrazumijeva prije svega da usmjeravanje na uređaju treba biti omogućeno i *seed* uređaj treba dostupan središnjem kontroleru. S druge strane, od sučelja povezanih s novim uređajima očekuje se da nemaju konfiguriranu IP adresu i da su linkovi koji ih povezuju linkovi drugog sloja OSI modela. Također, *seed* uređaji ne bi trebali biti povezani na neki drugi DHCP poslužitelj i naposljetku, potrebno je konfigurirati SNMP i akreditive uređaja.

[12] *Loopback* sučelje je virtualno sučelje koje je uvijek dostupno. Zbog navedenih karakteristika često se koristi za pronalazak problema ili za identifikaciju uređaja.

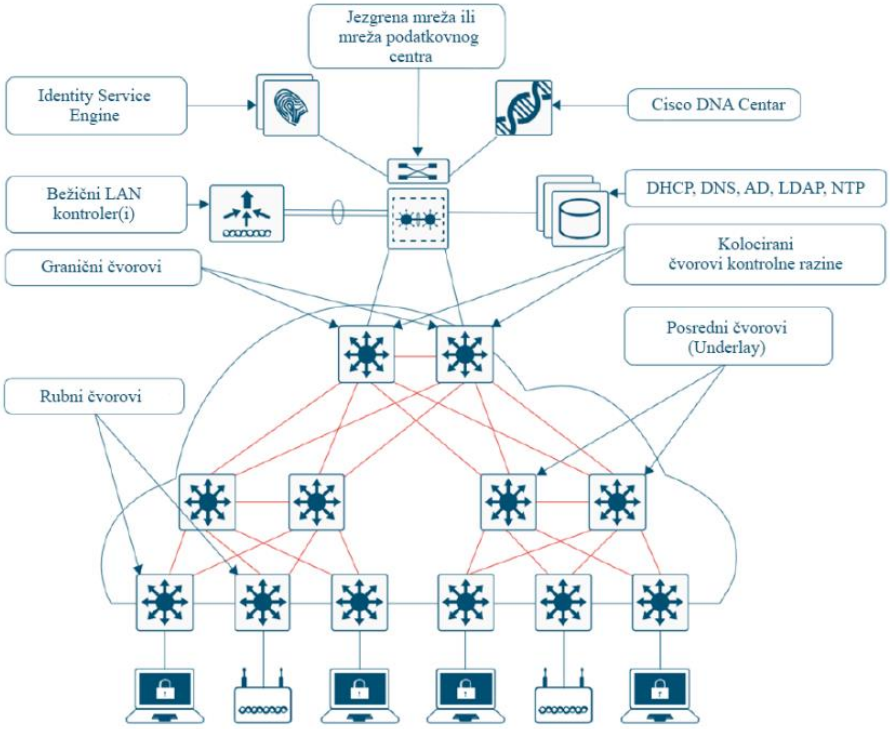
4. Arhitektura softverski definiranog pristupa u žičanim i bežičnim mrežama

SD-*Access* je rješenje koje olakšava IT transformaciju poboljšanjem preglednosti mrežnih događaja, definiranjem i implementiranjem politike temeljene na grupama, segmentacijom mreže za izoliranje prometa, smanjenjem rizika i vjerojatnosti pojave određenih prijetnji te ostvarivanjem dosljednosti u primjeni korisničkih politika. Ključni elementi koji omogućuju SD-*Access* rješenje su:

- Arhitektura temeljena na kontroleru
- *Network fabric* i
- Programibilna infrastruktura.

4.1. Elementi softverski definiranog pristupa

Prilikom prikaza arhitekture SD-*Access* rješenja, prije svega je bitno definirati različite elemente koji čine *fabric* topologiju, zatim navesti mogućnosti koje ti elementi pružaju i opisati njihovu interakciju. Unutar SD-*Access* rješenja, *fabric* topologiju čine neovisni set čvorova kontrolne razine (engl. *Control Plane Node*), rubni čvorovi (engl. *Edge Node*), posredni čvorovi (engl. *Intermediate Node*) i granični čvorovi (engl. *Border Nodes*). Integracija bežične mreže uključuje primjenu bežičnog LAN kontrolera (engl. *Wireless LAN Controller* - WLC) i bežične pristupne točke (engl. *Access Point* - AP) koji su prilagođeni za upotrebu unutar *fabric* topologije. Na slici 8 prikazano je rješenje za SD-*Access* koje je ponudio *Cisco* koje kombinira *Cisco* DNA Centar softver, *Identity Service Engine* (ISE) te žičnu i bežičnu funkcionalnost.



Slika 8 SD Access rješenje i prikaz elemenata fabric topologije. Izvor: [15]

Također, za međusobno povezivanje više *fabric* topologija koriste se tranzitne mreže kao IP *transit*, SD-WAN *Transit* i SD-*Access Transit* kojima se omogućuje kreiranje većih domena [15].

4.1.1. Čvorovi kontrolne razine

Čvorovi kontrolne razine koriste se kao centralizirane baze podataka, pri čemu imaju ulogu da prate klijente prilikom povezivanja na mrežu ili tijekom *roaminga*. Čvor kontrolne razine drugim mrežnim elementima omogućuje otkrivanje lokacije klijenata povezanih na mrežu koji bi inače trebali upotrebljavati mehanizam preplavljivanja i učenja [14]. Čvor kontrolne razine ažurira se izravno od strane rubnih čvorova (preklopnika pristupne razine) ako su u pitanju žičano povezani klijenti. S druge strane, WLC koji upravlja domenom bežične mreže razmjenjuje informacije i ažurira kontrolne čvorove prilikom povezivanja klijenata ili *roaminga* između AP-ova.

Funkcionalnost čvorova kontrolne razine moguće je migrirati na granične čvorove ili se mogu koristiti dedicirani čvorovi prilikom čega se preporučuje upotreba više njih zbog redundancije [15].

4.1.2. Granični čvorovi

Granični čvorovi koriste se kao *gateway* između SD-*Access fabric* topologije i tradicionalnih mreža drugog i trećeg sloja OSI modela ili pak nekih drugih *fabric* okruženja te su odgovorni za translacije konteksta vezanog za identitet i mapiranje korisnika i uređaja [15]. Granični čvor predstavlja točku gdje čvorovi kontrolne razine različitih *fabric* okruženja razmjenjuju informacije o dostupnosti i politiku [1].

Granični čvor omogućuje funkcionalnosti drugog i trećeg sloja OSI modela. Funkcionalnost drugog sloja omogućuje fleksibilnost dijeljenja istog mrežnog adresnog prostora unutar i izvan *fabric* topologije pri čemu se omogućuje međusobna komunikacija između klijenata koji se nalaze u tradicionalnom mrežnom okruženju i onima u *fabric* okruženju [14].

Prema funkcionalnostima trećeg sloja, granični čvorovi dijele se na one interne i eksterne. Granični čvorovi koji nisu eksplicitno definirani kao *default* čvorovi, predstavljaju interne čvorove i reklamiraju kontrolnoj razini skup poznatih mrežnih područja kao što su podatkovni centri ili dijeljeni poslužitelji. S druge strane, *default* ili eksterni čvorovi koriste se za dohvaćanje nepoznatih odredišta, poput onih na Internetu. Eksterni čvorovi predstavljaju efikasan mehanizam koji pruža *default* izlaz prema ostalim virtualnim mrežama bez potrebe uvođenja eksternih ruta [15].

4.1.3. Rubni čvorovi

Rubni čvorovi koriste se kao točke koje povezuju klijentske uređaje na SD-*Access fabric*, prilikom čega provode enkapsulaciju i dekapsulaciju te prosljeđuju promet od i prema klijentskim uređajima [14]. Rubni čvorovi nalaze se na samom rubu *fabric* topologije i predstavljaju prvu točku na koju se klijentski povezuju i na kojima se primjenjuje politika mreže. Rubni čvorovi

ekvivalentni su preklopnicima na pristupnom sloju u tradicionalnim mrežama. Kada se klijentski uređaji povežu na SD-*Access fabric*, rubni čvorovi vrše autentifikaciju upotrebom statične autentifikacije (mapiranje porta i odgovarajuće virtualne mreže ili grupe) ili dinamične autentifikacije putem 802.1X standarda (pridruživanjem identiteta klijenta odgovarajućoj virtualnoj mreži ili grupi) [1].

Za promet koji dolazi s prikvačenih klijentskih uređaja, rubni čvor upotrebom čvora kontrolne razine određuje odredište na koje će se promet proslijediti, zatim se dolazni paketi enkapsuliraju u odgovarajuće VXLAN zaglavlje zbog primjenjivanja određene politike i prosljeđivanja te se naposljetku i prosljeđuju prema odgovarajućem odredištu [1].

Također, jedan od zadataka ovog čvora jest i omogućavanje *Anycast Gateway* funkcionalnosti. Moguće je koristiti jedinstveni *gateway,* s jedinstvenom virtualnom IP/MAC adresu na svakom čvoru, zbog čega se omogućuje mobilnost klijenata i promjena rubnog čvora bez promjene *default gateway* informacija. Ova funkcionalnost pruža jednostavnost pri planiranju adresnog prostora i implementaciji istog [15].

4.1.4. Posredni čvorovi

Posredni čvorovi rade na trećem sloju OSI modela i koriste se za povezivanje rubnih te graničnih čvorova. U slučaju dizajna mreže s tri sloja (jezgreni, distribucijski i pristupni sloj), posredni čvorovi imaju funkcionalnost sličnu preklopnicima na distribucijskom sloju. Posredni čvorovi imaju za svrhu usmjeravanje i prijenos IP prometa unutar *fabric* mreže, a ne bave se enkapsulacijom i dekapsulacijom VXLAN zaglavlja, porukama kontrolne razine kao ni grupama klijenata

4.1.5. Bežični LAN kontroler

WLC za *fabric* okruženje integriran je s kontrolnom razinom i podržava AP-ove koji su i sami namijenjeni za *fabric* okruženje te koji su povezani na rubne čvorove. Osim što posjeduje

funkcionalnosti kao i tradicionalni WLC, kao što su pridruživanje klijenata na mrežu, upravljanje i konfiguracija AP-ova, upravljanje radiofrekvencijskim karakteristikama mreže te ostale slične operacije, WLC za *fabric* okruženje pruža dodatne funkcionalnosti koje omogućuju integraciju u *fabric* okruženje. Tako WLC za *fabric* okruženje sprema MAC adrese bežičnih klijenata tijekom njihovog pridruživanja na mrežu u bazu podataka koja se nalazi na kontrolnoj razini [1]. Ključna razlika između tradicionalnog i *fabric* WLC-a je u tome da *fabric* WLC nije aktivni sudionik u ulozi prosljeđivanja prometa na podatkovnoj razini. U ovom slučaju AP-ovi izravno prosljeđuju promet na rubne čvorove [15].

4.1.6. Bežična pristupna točka

AP se u tradicionalnim mrežama povezuje na rubni čvor, odnosno preklopnik i ima za svrhu povezivanje bežičnih klijenata na mrežu. AP-ovi za *fabric* okruženje također se povezuju izravno na rubni čvor. No, za razliku od prethodnog slučaja, preklopnik dalje prepoznaje AP kao posebnog žičanog klijenta zbog čega se uspostavlja posebna konfiguracija na portu i AP se pridružuje zasebnom *overlay* sloju. Tako se omogućuje pojednostavljeno upravljanje gdje se upotrebljava jedinstvena podmreža za sve AP-ove u *fabric* okruženju [15].

Kada se bežični klijenti povežu na *fabric* AP te se autentificiraju, WLC ažurira AP sa virtualnom mrežom i grupom klijenata koju je pružio ISE. Potom WLC registrira klijenta u bazu podataka na kontrolnom sloju. *Fabric* AP-ovi omogućuju distribuirano bežično prosljeđivanje u SD-*Access* arhitekturi enkapsulacijom prometa bežičnih klijenata u VXLAN zaglavlje i prosljeđivanje do izravno povezanog rubnog čvora gdje se obavlja dekapsulacija te primjenjuje mrežna politika nakon čega se ponovo enkapsulira i prosljeđuje dalje prema odredištu [14].

4.1.7. Identity Services Engine

ISE je proizvod tvrtke *Cisco Systems* i predstavlja platformu koja pruža klijentima siguran pristup mreži, a s druge strane administratorima pruža bolji pregled povezanih klijenata i

pokrenutih aplikacija. ISE predstavlja ključan dio u SD-*Access* mrežama za implementaciju mrežne politike, pružanje dinamičnog mapiranja klijenata s odgovarajućim grupama te jednostavniju implementaciju *end-to-end* sigurnosti [21]. ISE omogućuje upravljanje grupama, politikom, *Authentication, Authorization* i *Accounting* uslugama te profiliranjem krajnjih klijenata dok se orkestracija provodi na DNA softveru [15].

4.1.8. DNA Centar

DNA Centar je također proizvod tvrtke *Cisco Systems,* a predstavlja softversko rješenje koje pruža platformu s koje se omogućuje dizajniranje, odobravanje, praćenje i održavanje SD-*Access* rješenja. Također, osim pružanja automatizacije i mogućnosti dizajniranja, ovo rješenje podržava platformu za analitičke funkcionalnosti koje su neophodne za uspješno upravljanje takvom mrežom.

4.2. Slojevita arhitektura softverski definiranog pristupa

Network fabric je pojam koji se upotrebljava za opis topologije koja pruža žičanim i bežičnim mrežama programibilni *overlay*, omogućuje mrežnu virtualizaciju te odobrava fizičkoj mreži da ugosti jednu ili više logičkih mreža. Nadalje, ova topologija poboljšava kontrolu nad komunikacijom pružajući softverski definiranu segmentaciju i politiku na osnovi identiteta klijenta. *Network fabric* čine dva sloja, a to su fizički *underlay* i logički *overlay* slojevi [15].

4.2.1. SD-Access underlay

SD *Access underlay* čine fizički mrežni uređaji, kao što su usmjernici, preklopnici i WLC te tradicionalni usmjerivački protokoli trećeg sloja OSI modela pri čemu se *underlay* ne koristi za klijentski promet.

Svi mrežni elementi trebaju međusobno uspostaviti IPv4 konekciju što znači da se postojeća tradicionalna mreža može iskoristiti kao *underlay*. Iako se može koristiti bilo koja topologija i usmjerivački protokol, za osiguravanje odgovarajućih performansi, skalabilnosti i visoke dostupnosti, preporučuje se upotreba usmjerivačke pristupne topologije [14]. Ova vrsta topologije podrazumijeva da se ne upotrebljavaju linkovi između pristupnih preklopnika te da su svi linkovi usmjereni prema usmjernicima na trećem sloju OSI modela. Također, upotreba usmjerivačke pristupne topologije eliminira potrebu za protokolima kao što su STP, HSRP, VRRP i sličnih protokola što znatno pojednostavljuje mrežu i poboljšava toleranciju na pogreške u mreži. Upotreba logičke *fabric* topologije iznad odgovarajućeg *underlay* sloja pruža funkcionalnosti, kao što su ECMP, optimiziranu konvergenciju, jednostavnija implementacija, otkrivanje problema te naposljetku i upravljanje mrežom [22].

4.2.2. SD-Access overlay

SD *Access overlay* je logička, odnosno virtualizirana topologija smještena povrh fizičkog *underlay* sloja. Ovaj sloj pruža segmentiranje mreže na osnovu definirane politike, nudi mobilnost za žičane i bežične klijente te poboljšanu sigurnost. *Overlay* je potpuno automatiziran i u potpunosti je neovisan o modelu mreže ispod [1].

Overlay se ponaša kao baza podataka koja prati klijente povezane na *underlay*. *Overlay* se brine o registriranju svih povezanih klijenata u bazu podataka i prati na koji su rubni čvor povezani. S druge strane odgovara na upite drugih mrežnih elemenata o lokaciji klijenta na mreži. Također, kada se klijent premjesti s jedne lokacije na drugu, *overlay* se brine da se sav promet namijenjen tom klijentu preusmjeri na njegovu trenutnu lokaciju [15].

4.3. Integracija bežične mreže sa softverski definiranim pristupom

SD-*Access* pruža dvije mogućnosti kada je u pitanju integracija bežične mreže sa SD-*Access* mrežom [15]:

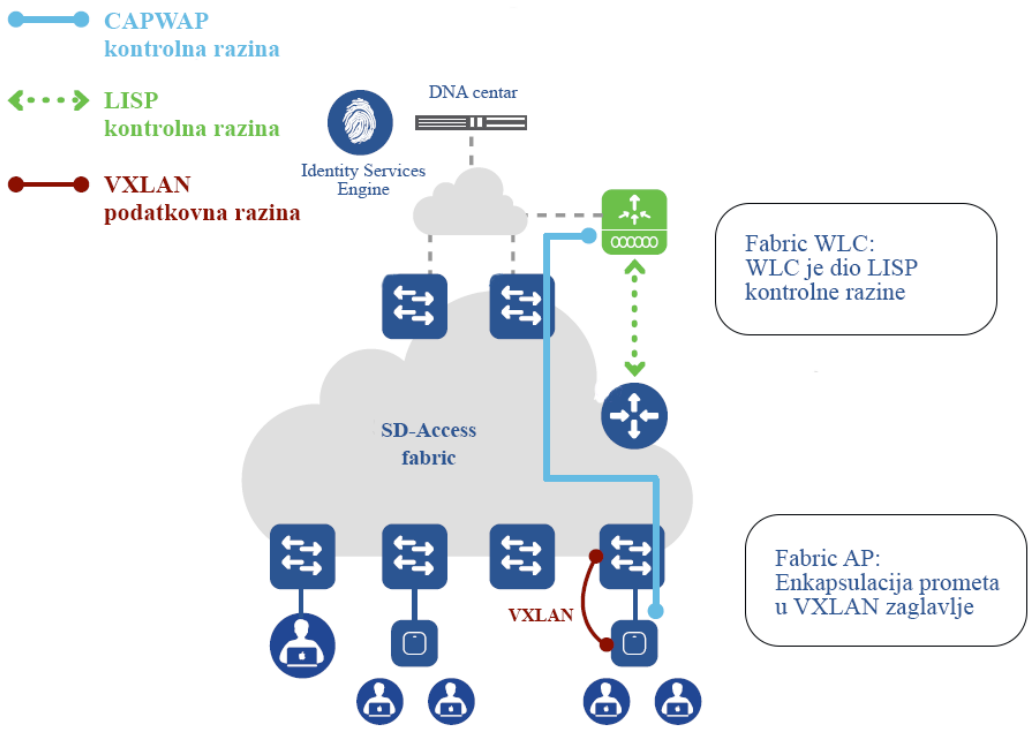
- SD-*Access* implementacija bežične mreže – potpuna integracija bežičnog prometa sa SD-*Access* mrežom i
- *Over-the-top* implementacija bežične mreže – SD-*Access* predstavlja transportnu mrežu za bežični promet.

4.3.1. SD-Access implementacija bežične mreže

U bežičnoj SD-*Access* implementaciji, žična i bežična mreža dio su jedinstvene integrirane infrastrukture i jednako se ponašaju po pitanju povezivanja, mobilnosti i primjeni politike mreže. Kada je u pitanju integracija sa kontrolnom razinom SD-*Access* mreže, WLC namijenjen za *fabric* topologiju informira čvor kontrolne razine kada se bežični klijenti pridružuju, vrše *roaming* ili prekidaju vezu [14]. Klijenti se registriraju prema MAC adresi te SGT i VN oznakama. Na ovaj način čvor kontrolne razine uvijek posjeduje jednake informacije o svim žičanim i bežičnim klijentima [15].

Također, WLC je zadužen za upravljanje i kontrolu AP-ova kao što je bio i slučaj u tradicionalnim bežičnim mrežama. Značajna je razlika u tome da klijentski promet ne prolazi *Control and Provisioning of Wireless Access Points*[13] (CAPWAP) enkapsulaciju [15]. WLC izdaje instrukcije *fabric* AP-ovima za formiranje VXLAN *overlay* tunela do susjednih rubnih čvorova. VXLAN tunel prenosi informacije vezane za segmentaciju i politiku mreže čime se omogućuje funkcionalnost jednaka kao i kod žičanih klijenata [14]. Prometni tok od bežičnog klijenta tako koristi najoptimalniji put preko *fabric* mreže do odredišta, umjesto da se klijentski tok šalje prvo na WLC, a tek onda prosljeđuje dalje [15].

[13] CAPWAP je protokol koji omogućuje konfiguraciju i upravljanje AP-ova i bežičnih mreža od strane WLC-a.

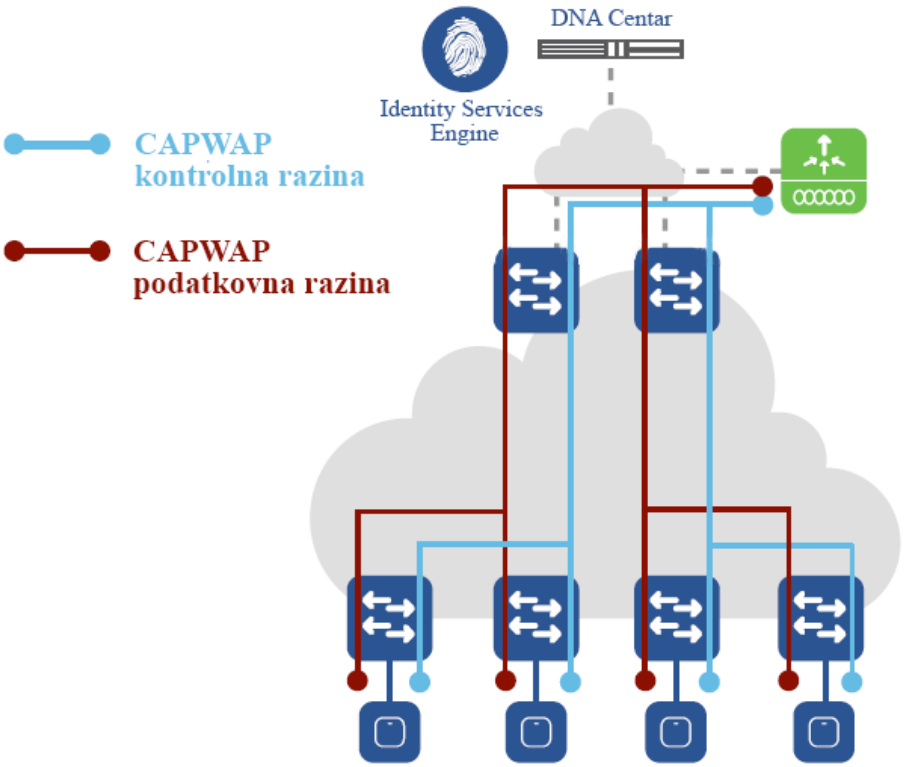


*Slika 9 Prikaz SD-*Access *implementacija bežične mreže. Izvor:* [14]

WLC može biti izravno povezan na granični čvor ili može biti smješten u podatkovnom centru. U slučaju kolociranog WLC-a, prefiks IP adrese podmreže u kojoj se WLC nalazi treba biti reklamiran na *underlay* sloju. Također, WLC je moguće smjestiti na smjestiti na isti uređaj na kojemu je i granični čvor. S druge strane, AP-ovi mogu biti povezani izravno na rubne ili na proširene čvorove [15]. SD-*Access* implementacija bežične mreže prikazana je slikom 9.

4.3.2. Over-the-top implementacija bežične mreže

SD-*Access* podržava implementaciju tradicionalnih AP-ova i WLC-ova te se u tom slučaju infrastruktura bežične mreže implementira povrh SD-*Access fabric* arhitekture. *Over-the-top* implementacija bežične mreže prikazana je slikom 10.



Slika 10 Pikaz Over-the-top implementacija bežične mreže. Izvor: [14]

Over-the-top rješenje podržava i dalje upravljanje adresnim prostorom, pojednostavljenu konfiguraciju i otkrivanje problema. U centraliziranom modelu, WLC i AP-ovi su smješteni u istom okruženju te je omogućeno povezivanje WLC-a na jezgrenu ili mrežu podatkovnog centra. Prometni tokovi između bežičnih klijenata i ostatka lokalne mreže enkapsuliraju se upotrebom CAPWAP protokola između WLC-a i AP-ova [15].

5. Pregled Zero Touch Provisioning modela

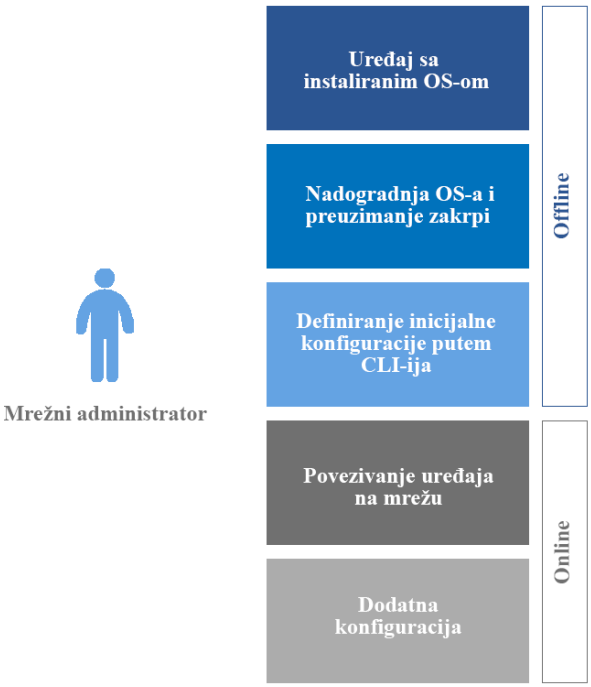
Suvremene mreže danas se susreću s izazovima novih poslovnih zahtjeva u vidu neometanog pristupa i podržavanja aplikacija koje postaju sve složenije i dinamičnije u svom razmjeru pri čemu se javlja potreba za korištenjem naprednih mrežnih usluga i distribuiranih poslužitelja. Dodavanjem novih usluga uz popratnu mrežnu infrastrukturu s ciljem zadovoljenja novih poslovnih zahtjeva, mrežnim se administratorima uz postojeće svakodnevne zadatke upravljanja i optimizacije znatno povećava opseg posla. Kako bi se omogućilo zadovoljenje klijentskih zahtjeva uz pojednostavljenje postupka pružanja mrežnih usluga i implementacije mrežne infrastrukture razvijen je *Zero Touch Provisioning* (ZTP) model [3].

ZTP model zasniva se na SDN konceptu i primjenjuje se u softverski definiranim mrežama. Holistički pristup definiciji ZTP modela sugerira da bi mrežni uređaji koji se povezuju na mrežu trebali biti automatski konfigurirani. Detaljnije, ZTP model bi se prema nekim autorima mogao definirati kao automatizacijsko rješenje razvijeno u svrhu smanjivanja vjerojatnosti pojave grešaka i uštede vremena mrežnim administratorima pri implementaciji nove mrežne infrastrukture [21, 22, 23]. ZTP model kao automatizacijsko rješenje nije ograničen samo na jedan uređaj, naime ovaj model pruža mogućnost *multipoint* automatizacije konfiguracije. Tako je umjesto upotrebe sučelja naredbenog retka (engl. *Command-Line Interface* - CLI) za konfiguraciju svakog uređaja zasebno moguće koristiti automatizacijske alate za preuzimanje operativnog sustava (OS), zakrpa i konfiguracija s distribuiranih poslužitelja [2].

5.1. Postupak izvršavanja Zero Touch Provisioning procesa

U tradicionalnim mrežama upravljanje mrežnim uređajima uglavnom se vrši upotrebom CLI-ja. Mrežni bi uređaji u tradicionalnim mrežama došli s već instaliranim OS-om, a mrežni bi administratori upotrebom CLI-ja pripremili uređaj za implementaciju u mrežu. Kada se uređaj pokreće po prvi put, uglavnom se ne povezuje odmah na mrežu, budući da mrežni administrator prije svega vrši provjeru verzije OS-a, preuzima zakrpe i nadogradnje. Nakon završetka početne provjere i nadogradnje sustava unosi se inicijalna konfiguracija parametara, kao što su informacije

za autentifikaciju administratora i korisnika, IP adresa namijenjena za upravljanje uređajima, adresa za *default gateway* i osnovne mrežne usluge (DHCP, *Network Time Protocol* i slično). Nakon što se verificira OS i inicijalna konfiguracija, uređaj je napokon moguće implementirati u mrežu i nakon čega je moguće konfigurirati uređaj lokalno ili udaljenim pristupom. Dijagram toka implementacije mrežnog uređaja na tradicionalan način prikazan je slikom 11.

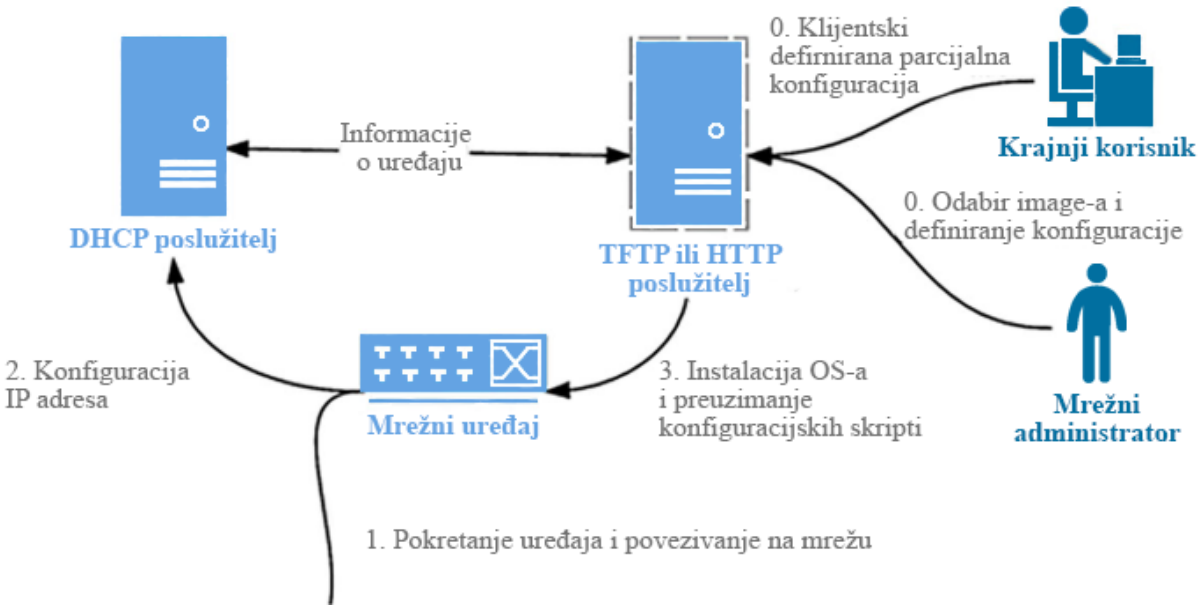


Slika 11 Dijagram toka tradicionalnog načina implementacije mrežnog uređaja. Izvor: [3]

Pri upotrebi ZTP pristupa za konfiguraciju mrežnih uređaja, administrator zaprima uređaj i fizički ga instalira u mrežni ormar i provodi kabliranje te povezuje uređaj na mrežu. Nakon što se poveže na mrežu, uređaj se pokreće i uz pomoć standardnih mrežnih protokola pribavlja sve što je potrebno za preuzimanje konfiguracije. Izvođenje ZTP procesa može se opisati kroz nekoliko koraka (slika 12):

1. Pokretanje mrežnog uređaja i povezivanje na mrežu.
2. Uređaj locira DHCP poslužitelj nakon čega mu se pridružuje IP adresa, *default gateway* adresa i po potrebi adresa DNS servera.

3. DHCP poslužitelj će uređaju također pružiti IP adresu *Trivial File Transfer Protocol* (TFTP) poslužitelja ili *Uniform Resource Locator* (URL) za pristup *HyperText Transfer Protocol* (HTTP) poslužitelju s kojih uređaj preuzima *system image* i konfiguracijsku skriptu koja se potom izvršava na mrežnom uređaju.



Slika 12 Izvođenje ZTP procesa po koracima. Izvor: [24]

Prije početka ZTP procesa potrebno je odgovarajuće *system image*-e i konfiguracijske skripte postaviti na TFTP ili HTTP poslužitelj. Također, postoji mogućnost upotrebe *web* sučelja za postavljanje konfiguracijskih skripti na poslužitelje, čime bi se dodatno olakšao proces implementacije novih mrežnih uređaja. Ovime se otvara mogućnost pružanja kontrole krajnjem korisniku nad mrežnom infrastrukturom kada se zahtijeva određena usluga [3].

DHCP poslužitelj može se konfigurirati tako da mrežni uređaj preuzme konfiguracijske datoteke ili skripte s TFTP ili HTTP poslužitelja. ZTP je na uređajima podržan na upravljačkim i na mrežnim portovima [25, 26]. Kada se uređaj poveže na mrežu zaprimi IP adresu TFTP poslužitelja ili URL za pristup HTTP poslužitelju i datotečni put do konfiguracijske datoteke, zbog čega je na DHCP poslužitelju potrebno konfigurirati dvije opcije:

- Opcija 150 – Sadrži listu IP adresa koje ukazuju na TFTP ili HTTP poslužitelj i
- Opcija 67 – Sadrži datotečni put do konfiguracijske datoteke ili skripte na TFTP ili HTTP poslužitelju.

Različiti proizvođači imaju tek neznatne razlike pri implementaciji ZTP procesa, za usporedbu su uzeti *Cisco Systems, Juniper Networks* i *Arista*, kao neki od najpoznatijih proizvođača mrežne opreme. Svi navedeni proizvođači podržavaju preuzimanje *system image*-a, konfiguracijskih datoteka i skripti, pri čemu se lokacija resursa pribavlja od strane DHCP poslužitelja. Također, svi navedeni proizvođači nude pristup njihovom *cloudu* za podršku pri prvom pokretanju uređaja. Bitna razlika je tek pri izvršavanju skripte gdje svaki proizvođač nudi vlastito okruženje na kojemu se konfiguracijska skripta izvršava [25, 27, 28]. Tako će se za detaljan opis procesa izvršavanja konfiguracijske skripte koristiti primjer u slučaju mrežnih uređaja tvrtke *Cisco Systems* koja predstavlja jednu od najvećih tvrtki u tom području prema ostvarenim prihodima u 2018. godini [29].

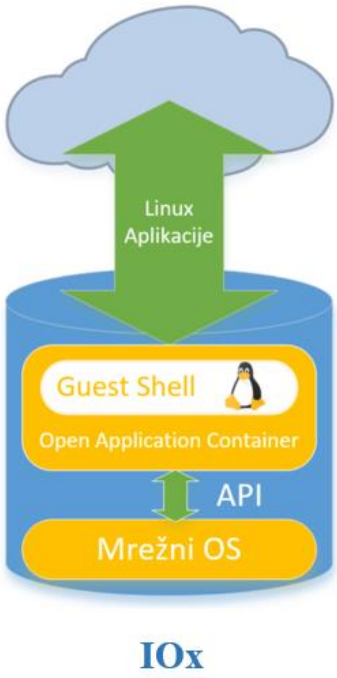
5.2. Guest Shell okruženje

Guest Shell je virtualizirano okruženje koje se temelji na *Linux*-u, a razvijeno je za pokretanje prilagođenih *Linux* aplikacija, uključujući i *Python* skripte za kontrolu i upravljanje *Cisco* mrežnim uređajima. Također, bitna funkcionalnost *Guest Shell* okruženja jest omogućavanje automatizacije konfiguracije pri prvom pokretanju uređaju. Nadalje, upotrebom *Guest Shell* okruženja moguće je instalirati, nadograditi i upravljati *Linux* aplikacijama trećih strana te je namijenjeno prvenstveno za određene alate za upravljanje mrežnim uređajima [30].

Guest Shell dijeli *kernel* s operativnim sustavom mrežnog uređaja, u ovom slučaju je to IOS XE. IOS XE je inačica *Cisco Internetworking Operating System* koja omogućuje programibilna sučelja i podatkovne modele. Ova inačica temelji se na *Linuxu* i pruža distribuiranu softversku arhitekturu koja oslobađa IOS procese od mnogih zadaća i posjeduje zasebnu kopiju IOS-a koja se pokreče kao skup zasebnih procesa.

Korisnicima je omogućen pristup *Guest Shell* okruženju i dopušteno je uređivati skripte te nadograđivati softverske pakete, no nije moguće izmijeniti datotečne sustave i utjecati na procese

hosta. *Guest Shell* softverskim jedinicama upravlja IOx. IOx predstavlja aplikacijski *framework*, odnosno platformu koja pruža hosting mogućnosti za različite tipove aplikacija na *Cisco* mrežnim platformama, a *Guest Shell* predstavlja jednu takvu aplikaciju (slika 13) [30].



Slika 13 Pregled IOx platforme. Izvor: [31]

Unutar *Guest Shell* okruženja *Cisco* uređaji podržavaju *Python* verziju 2.7. Mogućnost *Python* skriptiranja pruža programski pristup CLI-ju uređaja za obavljanje različitih zadataka, između ostalog i provedbu ZTP procesa.

Pri izvođenju ZTP procesa najprije se omogućuje IOx koji upravlja raznim aplikacijama, između ostaloga i *Guest Shell* okruženjem. Idući korak predstavlja uspostavljanje *Virtual Port Group* (VPG) čime se pruža veza s *Guest Shell* kontejnerom. Nadalje, za omogućavanje komunikacije s *Guest Shell* kontejnerom potrebno je konfigurirati *Network Address Translation*[14] (NAT) za mapiranje jednog adresnog prostora na drugi, u ovom slučaju potrebno je mapirati VPG

[14] NAT proces koji omogućuje privatnim RFC 1918 adresama da pristupe Internetu. NAT omogućuje translaciju jedne ili više privatnih IP adresa u jednu ili više globalnih (jedinstvenih) IP adresa.

adresu na adresu fizičkog porta. Posljednji korak predstavlja samo omogućavanje *Guest Shell* okruženja nakon čega je moguće izvršiti skriptu u *Guest Shell Python* okruženju.

5.3. Python API

Python je interpretirani i objektno-orijentirani programski jezik visoke razine s dinamičnom semantikom. Podatkovne strukture visoke razine u kombinaciji s dinamičkim tipkanjem i povezivanjem pogodne su za brzi razvoj aplikacija kao i za skriptiranje. *Python* podržava module i pakete što potiče modularnost i ponovnu upotrebu koda. *Python* interpretator i opsežna standardna biblioteka dostupni su u izvornom ili binarnom obliku za sve važnije platforme [25].

Guest Shell, kao što je već spomenuto, podržava *Python* i omogućuje izvršavanje *Python* naredbi te skripti kojima se ostvaruje programski pristup CLI-ju mrežnog uređaja. Ova funkcionalnost *Guest Shell* okruženja omogućuje provedbu ZTP procesa, ali raznih drugih zadataka [25].

Guest Shell podržava *Python* verziju 2.7 u interaktivnom i ne-interaktivnom načinu rada. Interaktivni način rada omogućuje korisniku izvršavanje pojedinačnih *Python* funkcija za konfiguraciju uređaja. S druge strane, ne-interaktivni način rada omogućuje izvršavanje *Python* skripte koja se nalazi na lokaciji na uređaju koja je dostupna *Guest Shell* okruženju, što je obično *flash* memorija uređaja [25].

5.4. Python CLI modul

Za konfiguraciju *Cisco* mrežnih uređaja uglavnom se koristi CLI. CLI predstavlja sučelje koje se zasniva na tekstu. CLI je podijeljen u više razina gdje svaka razina nudi zaseban set naredbi za konfiguraciju, nadgledanje i održavanje operacija mrežnog uređaja. Upotrebom određenih naredbi moguće je prelaziti između različitih razina.

Kada se započne sesija na mrežnom uređaju, korisnik se nalazi na *user* EXEC razini koja je ograničena s brojem naredbi i koja dozvoljava provođenje zadataka koji ne utječu na konfiguraciju uređaja i ne otkrivaju cjelokupno stanje uređaja. Za pristup ostalim naredbama za prikaz konfiguracijskog stanja mrežnog uređaja potrebno je pristupiti *privileged* EXEC razini - drugi stupanj EXEC razine za koji je obično potrebno unijeti lozinku. S ove razine moguće je pristupiti *global configuration* razini s koje je moguće utjecati na generalnu konfiguraciju uređaja ili pristupiti specifičnim konfiguracijskim razinama.

Cisco pruža *Python* modul koji omogućuje izvršavanja CLI naredbi u *Guest Shell Python* okruženju. U *Python* CLI modulu dostupno je šest funkcija za izvršavanje različitih CLI naredbi i dodatna *help()* funkcija koja prikazuje značajke *Cisco* CLI modula. Za upotrebu ovih funkcija potrebno je prije svega uvesti modul naredbom *import cli modul.* Argument ovih funkcija jest *string* određene CLI naredbe. Dostupne su sljedeće funkcije [25]:

- **cli.cli(***naredba***)** – Funkcija uzima bilo koju IOS naredbu kao argument, provodi se parsiranje[15] i naredba se izvršava, nakon čega se vraća rezultirajući tekst. Ako je predana neispravna naredba, javlja se *Python* iznimka.
- **cli.clip(***naredba***)** – Ova funkcija se ponaša isto kao i prethodna, osim što se rezultirajući tekst ispisuje. Funkcije *cli* i *clip* pružaju mogućnost prosljeđivanja više naredbi.
- **cli.execute(***naredba***)** – Ova funkcija pojedinačno izvršava naredbe *user* i *privileged* EXEC razine i vraća rezultat izvršavanja.
- **cli.executep(***naredba***) –** Kao i prethodna, ova funkcija izvršava pojedinačne naredbe EXEC razina, no rezultat izvršavanja funkcije ispisuje se za razliku od prethodnog slučaja gdje se rezultat vraća.
- **cli.configure(***naredba***)** – Ova funkcija obavlja konfiguraciju mrežnog uređaja prema predanim naredbama u argumentu funkcije. Funkcija vraća listu koja sadrži predanu naredbu, uspjeh izvršavanja naredbe i informacije o pogreški ako ona postoji. Moguće je predati niz naredbi odvojenih zarezom.
- **cli.configurep(***naredba***)** – Ova funkcija ponaša se identično kao i prethodna, osim što ispisuje rezultat izvršavanja funkcije.

[15] Razrješavanje niza koda ili teksta u njegove elemente kako bi se utvrdilo jesu li su u skladu s definiranim uvjetima.

6. Razvoj okruženja Zero Touch Provisioning modela

Proces razvoja okruženja u kojemu će se prikazati funkcionalnosti ZTP modela zahtjeva pristup po koracima. Prvo, potrebno će biti definirati mrežnu topologiju u kojoj će se pokrenuti ZTP proces. Drugo, slijedi definiranje konfiguracije mrežnih uređaja i poslužitelja iz prethodno uspostavljene topologije mreže. Treće, razvoj skripte u programskom jeziku *Python* koja će se izvršavati na IOS XE podržanom preklopniku. Četvrto, slijedi povezivanje predmetnog uređaja na mrežu i pokretanje. Naposljetku slijedi komentiranje rezultata izvršavanja ZTP procesa.

Za razvoj okruženja ZTP modela korišten je softver *Emulated Virtual Environment – Next Generation* (EVE-NG), verzija 2.0.3-110. EVE-NG je softver za emulaciju mreže koji podržava opremu više različitih proizvođača. Ovaj emulacijski softver pruža alate za testiranje raznih mrežnih uređaja, međusobno povezivanje, ali i povezivanje s fizičkim uređajima. Također, ovaj softver podržava *system image* za Cisco IOS XE uređaje što ga čini pogodnim za upotrebu pri razvoju okruženja za provedbu ZTP procesa.

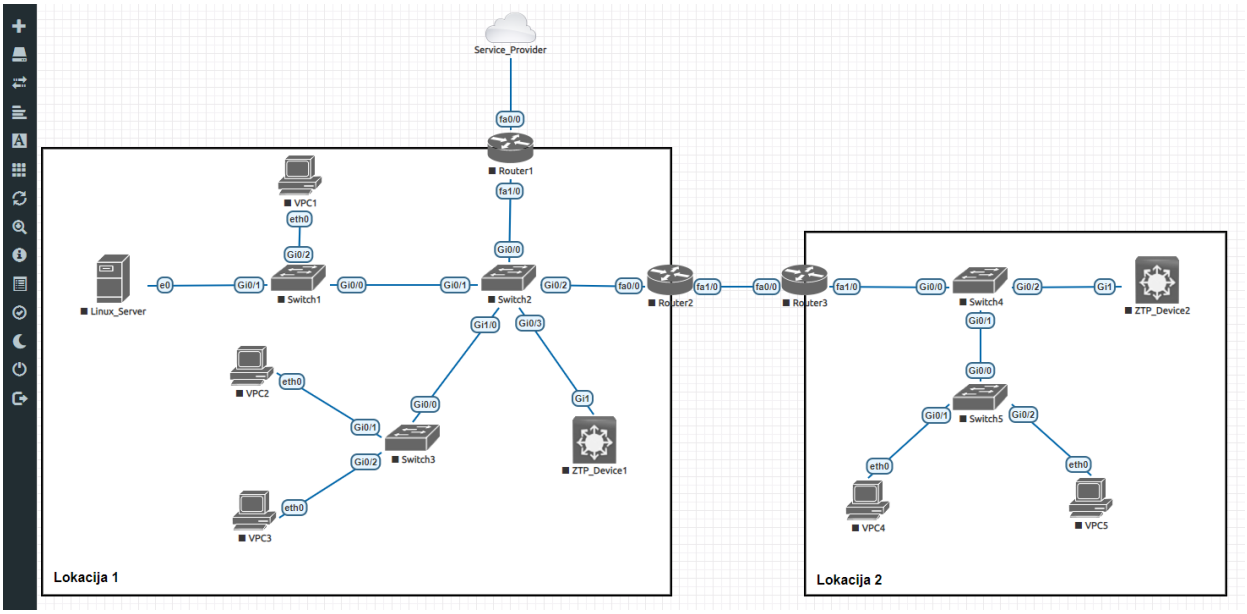
6.1. Razvoj generalne topologije mreže

Testiranje ZTP procesa provest će se u emuliranoj mreži manjeg poslovnog okruženja. Topologija mreže korištene za testiranje ZTP procesa prikazana je slikom 14. Za dizajniranje topologije ove mreže slijeđene su najbolje prakse za dizajniranje manjeg kampusa koje zagovaraju *Cisco* i *Juniper* [32, 33]. Koncept mreže za kampuse podrazumijeva povezivanje više LAN-ova koji su rasprostranjeni na nekom užem geografskom području. Od takve mreže očekuje se da omogući komunikaciju između klijenata koji se nalaze u različitim zgradama ili lokacijama kao i pristup klijentima prema Internetu.

Za prikaz ZTP procesa definirane su dvije odvojene fizičke lokacije, *Lokacija 1* i *Lokacija 2*, koje su povezane određenim tipom WAN konekcije. Na taj način klijentima je omogućena međusobna komunikacija. Pristup vanjskim mrežama, odnosno Internetu, omogućen je usmjernikom koji je povezan na mrežu pružatelja usluga.

Topologija slijedi hijerarhijski dizajn mreže s dva sloja, a to su pristupni i distribucijski, za kojeg se koristi još i naziv *Collapsed Core*. Ovakav način dizajniranja mreže omogućuje svakom mrežnom elementu da obavlja određenu ulogu u mreži [32]. Klijenti pristupaju mreži putem pristupnih preklopnika, u ovom slučaju to su čvorovi *Switch1*, *Switch3* i *Switch5*. Nadalje, ti pristupni preklopnici međusobno su povezani u agregacijskoj točki koju čini distribucijski preklopnik, a to je čvor *Switch2* na *Lokaciji 1* i čvor *Switch4* na *Lokaciji 2*. Na *Lokaciji 1* distribucijski preklopnik povezuje se dalje na usmjernik, odnosno čvor *Router1*, koji provodi usmjeravanje prometa prema drugim mrežama, odnosno na čvor *Router2*, koji predstavlja usmjernik povezan s usmjernikom na *Lokaciji 2*. Na *Lokaciji 2* nalazi se jedan usmjernik, čvor *Router3*, na koji se povezuje distribucijski preklopnik na ovoj lokaciji.

Preklopnici koji podržavaju ZTP proces, čvorovi *ZTP_device1* i *ZTP_device2*, nalazit će se u pristupnom sloju mrežne arhitekture i biti će povezani na preklopnike distribucijskog sloja, čvorove *Switch2* i *Switch4*. Također, u topologiji se nalaze još i *VPC* čvorovi koji predstavljaju klijente u mrežama.



Slika 14 Snimka zaslona radne okoline softvera EVE-NG koja prikazuje topologiju mreže korištene za testiranje ZTP procesa

Čvorovi usmjernika i pristupnika koriste odgovarajuće *system image*-e *Cisco* IOS operativnog sustava dok ZTP uređaji koriste *system image* XE 16.06.07. inačice *Cisco* IOS operativnog sustava. Za *Linux* poslužitelje koristi se *system image Ubuntu* 20.04. operativnog sustava.

6.2. Konfiguracija mrežnih elemenata

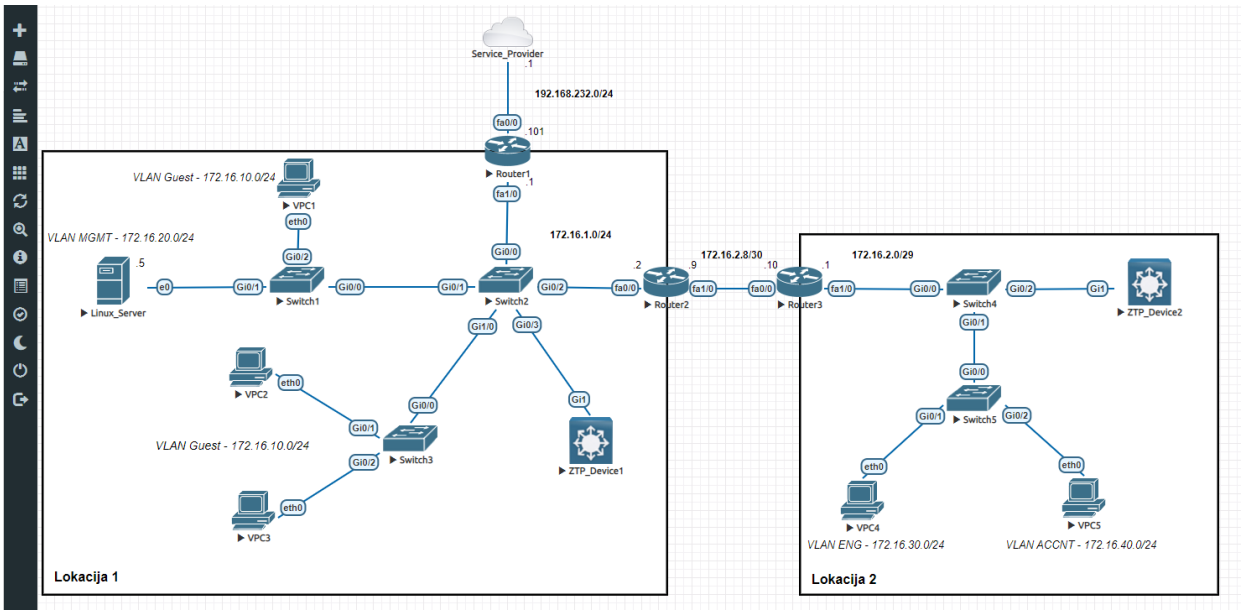
Nakon definiranja topologije mreže na *Lokaciji 1* i *Lokaciji 2*, slijedi konfiguracija mrežnih uređaja i poslužitelja na mreži. Prije svega potrebno je definirati uloge pojedinih slojeva kako bi se izvršila odgovarajuća konfiguracija. Tako su uređaji na pristupnom sloju konfigurirani kao preklopnici drugog sloja OSI modela koji prosljeđuju klijentski promet *trunk* portovima[16] prema distribucijskom sloju. Na distribucijskom sloju ponovo se koriste preklopnici koji rade na drugom sloju OSI modela. Oni su konfigurirani za prosljeđivanje silaznog prometa *trunk* portovima prema pristupnim preklopnicima i za prosljeđivanje uzlaznog prometa prema usmjernicima. U hijerarhijskom dizajnu s dva sloja, distribucijski sloj obnaša i ulogu jezgrenog sloja. Na tom sloju tako se nalaze i usmjernici trećeg sloja OSI modela koji pružaju pristup WAN-u ili Internetu.

Pristupni sloj pruža mrežni pristup širokom rasponu mrežnih uređaja i klijenata. Mrežni promet generiran od strane tih uređaja i klijenata obično ima različite sigurnosne i upravljačke zahtjeve, zbog čega ga je potrebno razdvojiti. Obično se u mrežama poslovnog okruženja razdvajanje prometa na drugom obavlja upotrebom VLAN-ova. Način razdvajanja klijenata obično ovisi o organizacijama gdje se primjenjuje. Tako je u ovoj topologiji korišteno razdvajanje prema vrsti prometa, odnosno poseban VLAN za upravljački promet, zatim za različite odjele u organizaciji kao što su računovodstvo i inženjerski dio te poseban VLAN za vanjske klijente. Na preklopnicima su tako konfigurirani pristupni portovi za različite vrste klijenata i *trunk* portovi između preklopnika te na linkovima s usmjernicima.

Za omogućavanje komunikacije klijenata različitih VLAN-ova definiran je *default gateway* na usmjernicima *Router1* na *Lokaciji 1* i *Router3* na *Lokaciji 2*. Za usmjeravanja unutar mreže korišten je protokol usmjeravanja na osnovu stanja mreže koji obično pružaju bolju skalabilnost i kraće vrijeme konvergencije u odnosu na protokole vektora udaljenosti. Tako je konkretno korišten

[16] *Trunk* portovi se konfiguriraju za prijenos podatkovnih tokova više VLAN-ova.

protokol *Open Shortest Path First* (OSPF), koji je ujedno i jedan od najčešće korištenih protokola usmjeravanja [33]. Kako bi se omogućio pristup Internetu definirana je i *default* ruta na usmjerniku *Router1* koja je potom reklamirana putem OSPF protokola ostalim usmjernicima u mreži. Također, kako bi se omogućio pristup Internetu potrebno je definirati i NAT koji štiti privatni adresni prostor unutar mreže i mapira privatne adrese na javne, odnosno adrese koje podržavaju usmjeravanje na Internetu. Prikaz topologije mreže s oznakama domena drugog sloja OSI modela i s prikazom korištenih adresa trećeg sloja nalazi se na slici 15.



Slika 15 Snimka zaslona radne okoline softvera EVE-NG koja prikazuje topologiju mreže sa konfiguriranim mrežnim elementima, pridruženim IP adresama i definiranim VLAN-ovima

Za provedbu ZTP procesa potrebno je uspostaviti i odgovarajuće poslužitelje. Zbog toga je na *Lokaciji 1* podignut *Linux* poslužitelj s *Ubuntu* 20.04. operativnim sustavom koji dalje ugošćuje DHCP, TFTP i HTTP poslužitelje. DHCP poslužitelj uspostavljen na *Linuxu* predstavlja središnji DHCP poslužitelj u mreži i zadužen je za dodjelu IP adresa svim klijentima na mreži, ali i omogućuje opcije 150 i 67 za ZTP proces na *Lokaciji 1*. Konfiguracija DHCP poslužitelja prikazana je slikom 16.

```
subnet 172.16.1.0 netmask 255.255.255.248 {
range 172.16.1.100 172.16.1.254;
option bootfile-name "ZTP_python.py";
option tftp-server-name "172.16.20.5";
option subnet-mask 255.255.255.0;
option routers 172.16.1.1;
option broadcast-address 172.16.1.255;
default-lease-time 600;
max-lease-time 7200;
}

subnet 172.16.10.0 netmask 255.255.255.0 {
range 172.16.10.100 172.16.1.254;
option subnet-mask 255.255.255.0;
option routers 172.16.10.1;
option broadcast-address 172.16.10.255;
default-lease-time 600;
max-lease-time 7200;
}

subnet 172.16.20.0 netmask 255.255.255.0 {
range 172.16.20.100 172.16.20.254;
option subnet-mask 255.255.255.0;
option routers 172.16.10.1;
option broadcast-address 172.16.10.255;
default-lease-time 600;
max-lease-time 7200;
}

subnet 172.16.30.0 netmask 255.255.255.0 {
range 172.16.30.100 172.16.30.0254;
option subnet-mask 255.255.255.0;
option routers 172.16.30.1;
option broadcast-address 172.16.30.255;
default-lease-time 600;
max-lease-time 7200;
}

subnet 172.16.40.0 netmask 255.255.255.0 {
range 172.16.40.100 172.16.40.0254;
option subnet-mask 255.255.255.0;
option routers 172.16.40.1;
option broadcast-address 172.16.40.255;
default-lease-time 600;
max-lease-time 7200;
}
```

Slika 16 Prikaz konfiguracije DHCP poslužitelja na Linux poslužitelju

U svrhu prikaza konfiguracije DHCP poslužitelja za ZTP proces na *Cisco* mrežnom uređaju, na čvoru *Router3* konfiguriran je DHCP koji ima svrhu dodjele upravljačkih adresa mrežnim uređajima na *Lokaciji 2* i na kojemu je također definirana opcija 150 i opcija 67. Konfiguracija DHCP poslužitelja na *Cisco* uređaju prikazana je slikom 17.

```
!
ip dhcp excluded-address 172.16.2.1
!
ip dhcp pool ZTP
network 172.16.2.0 255.255.255.248
default-router 172.16.2.1
option 67 ascii /tftpboot/ZTP_python.py
option 150 ip 172.16.20.5
!
```

Slika 17 Prikaz konfiguracije DHCP poslužitelja na Cisco uređaju

Budući da su klijenti na mreži razdvojeni u različite *broadcast* domene, potrebno je bilo na mrežnim sučeljima usmjernika konfigurirati DHCP *helper* adresu koja omogućuje da klijenti pristupe DHCP poslužitelju na drugoj podmreži.

DHCP poslužitelj koristi opciju 150 za pružanje IP adrese TFTP poslužitelja klijentima. TFTP poslužitelj također se nalazi na *Linux*-u. Na TFTP poslužitelju smještena je *Python* skripta koju pri pokretanju ZTP procesa uređaj preuzima. Uz TFTP, na *Linuxu* je smješten i HTTP poslužitelj s kojega se preuzima konfiguracijska datoteka.

6.3. Razvoj Python skripte

Cisco je u svojoj IOS XE inačici operativnog sustava omogućio pristup *Python* okruženju unutar *Guest Shell* okruženja iz kojeg je moguće uređaju predavati naredbe. Za predavanje naredbi razvijen je *Python* CLI modul koji posjeduje šest funkcija za predavanje različitih vrsta naredbi. Zbog toga je skriptu potrebno napisati u *Python* programskom jeziku.

Skripta ima svrhu da na osnovu serijskog broja uređaja preuzme odgovarajuću konfiguracijsku datoteku za taj uređaj i primjeni tu istu. Cilj skripte je da priskrbi uređaju osnovnu konfiguraciju kao i konfiguraciju za udaljeni pristup uređaju putem protokola kao što su *Telnet* i *Secure Shell* (SSH). Razvijena skripta prikazana je isječkom prikazanim na slici 18.

```
import re
from cli import cli, clip, configure, configurep, execute, executep
import json

print '\n\n******ZTP is beginning!******\n'

def get_serials():

    #Parsiranje serijskog broja

    print '\nSerialNo parsing...\n'
    inventory = cli('show inventory')
    invetoryList = inventory.split('\n\n')
    for chassis in invetoryList:
         if (re.search('NAME: "Chassis"',chassis)):
             match = re.findall('SN:\s(.*)', chassis)
    serialNo = match[0]
    print '\nSerial number: {}\n'.format(serialNo)
    return serialNo

def get_config():

    #Dohvacanje konfiguracije sa  HTTP servera

    url = 'http://172.16.20.5/device_config.json'

    print '\nDownloading configuration from URL: {}'.format(url)
    clip('copy http://172.16.20.5/device_config.json flash')
    config = cli('more flash:device_config.json')
    json_config = json.loads(config)
    print '\nConfiguration commands: Collected!\n'
    return json_config

def configure_device(config, serial):

    #Konfiguracija

    print '\nConfiguring device...\n'
    found = False
    for line in config:
        if line["SN"] == serial:
            device_config = line
            if 'hostname' in device_config and device_config['hostname'] is not None:
                print 'hostname: ' + device_config['hostname']
            if 'ipaddress' in device_config and device_config['ipaddress'] is not None:
                print 'ip address: ' + device_config['ipaddress']
            if 'netmask' in device_config and device_config['netmask'] is not None:
                print 'netmask: ' + device_config['netmask']
```

```
            print '\n'
            configurep(['hostname {}'.format(device_config['hostname'])])
            configurep(['int loopback1', 'ip add {} {}'.format(device_config['ipaddress'],
                        device_config['netmask']), 'no shut', 'end'])
            configurep(['line con 0', 'logging synchronous', 'no exec-timeout', 'end'])
            configurep(['ip domain-name domain.com'])
            configurep(['crypto key generate rsa'])
            configurep(['username User password admin'])
            configurep(['enable secret {}'.format(device_config['secret'])])
            configurep(['line vty 0 4', 'transport input ssh', 'login local'])
            found = True
            break
    if found == True:
        print '\n\nConfiguration applied!'
    else:
        print '\n\nConfiguration not applied!'

serial = get_serials()
config = get_config()
configure_device(config, serial)

print '\n\n******ZTP is finished!******\n\n'
```

Slika 18 Python skripta namijenjena za izvođenje na IOS XE Cisco uređaju

Ideja ovog postupka jest da se kreira jedna skripta koju je moguće pokrenuti na više uređaja na mreži, a na određenom poslužitelju bit će dostupna konfiguracijska datoteka koju će skripta preuzeti, a koja će sadržavati posebne vrijednosti za svaki uređaj. Konfiguracijska datoteka bit će zapisana u lako razumljivom *JavaScript Object Notation* (JSON) formatu pri čemu će vrijednosti moći unijeti bilo tko, a ne samo developer. Na ovaj način moguće je konfigurirati veliki broj uređaja na jako učinkovit način.

Skripta uvozi tri *Python* modula, a to su *re*, *cli* i *json*. Modul *re* ili *Regular Expression* predstavlja niz znakova koji tvore uzorak za pretraživanje određenog *stringa*. Zatim *cli* modul koji se, kao što je već spomenuto, koristi za predavanje naredbi mrežnom uređaju iz *Python* okruženja. I naposljetku, *json* modul koji se koristi za rad sa JSON podacima.

Nadalje, u skripti su definirane tri funkcije za svaki korak konfiguracije uređaja, a koje se pozivaju na kraju skripte. Prva funkcija, odnosno *get_serials(),* ima zadatak da upotrebom *re* modula parsira odgovarajući serijski broj mrežnog uređaja koji se na istom dobije predavanjem definirane naredbe putem *cli* modula. Ova funkcija naposljetku vraća varijablu u kojoj je sadržan

traženi serijski broj. Zatim, funkcija *get_config()* posjeduje URL za pristup HTTP serveru na kojemu se nalazi konfiguracijska datoteka u JSON formatu, koja je prikazana isječkom na slici 19.

```
[
   {
        "SN": "9KZE846DMPS",
        "hostname": "ZTPDevice1",
        "ipaddress": "172.16.1.10",
        "netmask": "255.255.255.0",
        "secret": "ztpdevice1"
    },
    {
        "SN": "9EICYJKBWJG",
        "hostname": "ZTPDevice2",
        "ipaddress": "172.16.1.20",
        "netmask": "255.255.255.0",
        "secret": "ztpdevice2"
    },
    {
        "SN": "9B5KRA6AGNV",
        "hostname": "ZTPDevice3",
        "ipaddress": "172.16.2.2",
        "netmask": "255.255.255.248",
        "secret": "ztpdevice3"
    },
    {
        "SN": "9MCDBBGGRUL",
        "hostname": "Router40",
        "ipaddress": "172.16.2.3",
        "netmask": "255.255.255.248",
        "secret": "ztpdevice4"
    }
]
```

Slika 19 Prikaz konfiguracijske datoteke u JSON formatu

Nakon što se pribavi, provodi se parsiranje JSON datoteke. Parsiranjem JSON stringa odgovarajućim metodama dobije se *Python dictionary* tip podataka koji je moguće podijeliti na dva dijela: ključ i vrijednost, što ga čini pogodnim za dobivanjem vrijednosti za pojedine varijable. Funkcija na kraju vraća dobiveni tip podataka. Posljednja funkcija, odnosno *configure_device()*, uzima rezultate prethodnih funkcija i koristi ih za konfiguraciju uređaja. Funkcija prije svega na

osnovu dobivenog serijskog broja pretražuje konfiguraciju za dati uređaj. Kada se pronađe konfiguracija za dati serijski broj započinje provjera kojom se utvrđuje da su zahtijevane vrijednosti prisutne. Na samom kraju započinje upotreba funkcija *cli* modula kojom se predaju naredbe mrežnom uređaju, pri čemu se ispisuje rezultat i uspjeh izvođenja naredbe.

6.4. Proces izvršavanja ZTP modela

Nakon što se razvije mrežno okruženje i kreira skripta moguće je započeti s izvođenjem ZTP procesa. Uređaji koji podržavaju ZTP proces, na mrežnoj topologiji to su *ZTP_device1* i *ZTP_device2*, povezani su na odgovarajuće preklopnike distribucijskog sloja i pokrenuti.

Cisco uređaj započinje proces pokretanja provjerom određenih komponenti, kao što su procesor, *Random Access Memory* (RAM) i *Non-Volatile Random Access Memory* (NV-RAM). Nakon što se sprovedu testovi, *bootstrap* program se kopira iz *Read-OnlyMemory* (ROM) u RAM memoriju. Nakon kopiranja, procesor izvršava instrukcije iz *bootstrap* programa. Glavna zadaća *bootstrap* programa jest da locira *Cisco* IOS i učita ga u RAM. Zadnji korak pri pokretanju uređaja je provjera postojanja *startup* konfiguracije u NVRAM memoriji, u čijem slučaju postojanja se učitava u RAM te se konfiguracija primjenjuje.

ZTP proces započinje u onom trenutku kada uređaj ne pronađe nikakvu *startup* konfiguraciju u NV-RAM memoriji i zaprimi DHCP *offer* koja uključuje opciju 150, odnosno 67. Uređaj dobiva informacije o lokaciji TFTP poslužitelja i datotečnom putu prema skripti koja se nalazi na tom poslužitelju te preuzima skriptu i sprema je u *flash* memoriju. Zatim započinje omogućavanje *Guest Shell* okruženja, kao što je već opisano u poglavlju 5.3.. Proces započinje omogućavanjem IOx platforme, zatim slijedi konfiguracija VPG-a i NAT-a, a na kraju slijedi omogućavanje *Guest Shell* okruženja. Proces pokretanja uređaja i omogućavanja *Guest Shell* okruženja prikazan je logovima u isječku na slici 20.

```
Cisco IOS Software [Everest], Virtual XE Software (X86_64_LINUX_IOSD-UNIVERSALK9-M), Version
16.6.7, RELEASE SOFTWARE (fc2)
Technical Support: http://www.cisco.com/techsupport
Copyright (c) 1986-2019 by Cisco Systems, Inc.
Compiled Mon 23-Sep-19 14:33 by mcpre
*Jul 15 09:59:26.315: %CRYPTO-6-ISAKMP_ON_OFF: ISAKMP is OFF
*Jul 15 09:59:26.316: %CRYPTO-6-GDOI_ON_OFF: GDOI is OFF
*Jul 15 09:59:29.001: %AN-6-AN_ABORTED_BY_MANUAL_CONFIG_DETECTED: Autonomic disabled due to
detection of new configuration.
*Jul 15 09:59:29.002: %AN-6-AN_ABORTED_BY_CONSOLE_INPUT: Autonomic disabled due to User
intervention on console. configure 'autonomic' to enable it.
*Jul 15 09:59:29.010: %SYS-6-BOOTTIME: Time taken to reboot after reload =  351 seconds
*Jul 15 09:59:34.113: AUTOINSTALL: Obtain tftp server address (opt 150) 172.16.20.5
*Jul 15 09:59:34.113: PNPA: Setting autoinstall complete to true for 172.16.20.5
*Jul 15 09:59:34.721: [IOX DEBUG] Guestshell start API is being invoked

*Jul 15 09:59:34.722: [IOX DEBUG] provided idb is front-panel interface

*Jul 15 09:59:34.722: [IOX DEBUG] unable to retrieve primary dns for network interface "Gi1"

*Jul 15 09:59:34.723: [IOX DEBUG] unable to retrieve secondary dns for network interface "Gi1"

*Jul 15 09:59:34.723: [IOX DEBUG] Configuring NAT outside for supplied interface:
GigabitEthernet1
*Jul 15 09:59:35.059: [IOX DEBUG] Configuring VirtualPrivateGroup (31) interface

*Jul 15 09:59:35.686: %IOSXE-4-PLATFORM:  R0/0: kernel: dev->name [intsvc31]: dev_entry not
populated
*Jul 15 09:59:35.956: [IOX DEBUG] Setting up chasfs for iox related activity

*Jul 15 09:59:35.956: [IOX DEBUG] Setting up for iox pre-clean activity if needed

*Jul 15 09:59:35.965: [IOX DEBUG] Waiting for iox pre-clean setup to take affect

*Jul 15 09:59:35.968: [IOX DEBUG] Waited for 1 sec(s) for iox pre-clean setup to take affect

*Jul 15 09:59:35.974: [IOX DEBUG] Auto-configuring iox feature

*Jul 15 09:59:36.053: [IOX DEBUG] Waiting for CAF and ioxman to be up, in that order

*Jul 15 09:59:37.418: %UICFGEXP-6-SERVER_NOTIFIED_START:  F0: psd:  Server iox has been
notified to start
*Jul 15 09:59:38.723: %LINEPROTO-5-UPDOWN: Line protocol on Interface VirtualPortGroup31,
changed state to up
*Jul 15 09:59:40.213: [IOX DEBUG] Waiting for another 5 secs

*Jul 15 09:59:45.532: [IOX DEBUG] Waiting for another 5 secs

*Jul 15 09:59:53.727: [IOX DEBUG] Waiting for another 5 secs

*Jul 15 09:59:58.772: [IOX DEBUG] Waiting for another 5 secs

*Jul 15 10:00:03.790: [IOX DEBUG] Waiting for another 5 secs

*Jul 15 10:00:09.918: [IOX DEBUG] Waiting for another 5 secs

*Jul 15 10:00:15.570: [IOX DEBUG] Waiting for another 5 secs

*Jul 15 10:00:22.913: [IOX DEBUG] Waiting for another 5 secs
```

```
*Jul 15 10:00:27.937: [IOX DEBUG] Waiting for another 5 secs

*Jul 15 10:00:32.939: [IOX DEBUG] Waiting for another 5 secs

*Jul 15 10:00:37.947: [IOX DEBUG] Waiting for another 5 secs

*Jul 15 10:00:42.956: [IOX DEBUG] Waiting for another 5 secs

*Jul 15 10:00:47.961: [IOX DEBUG] Waiting for another 5 secs

*Jul 15 10:00:48.636: %VXE_VNIC_IF-4-DRIVER_NOT_SUPPORTED: Ignoring interface with address
1a:41:81:a8:0f:3d using unsupported veth driver.The process for the command is not re
*Jul 15 10:00:51.880: [IOX DEBUG] Waited for 67 sec(s) for CAF and ioxman to come up

*Jul 15 10:00:51.883: [IOX DEBUG] Validating if CAF and ioxman are running

*Jul 15 10:00:51.889: [IOX DEBUG] CAF and ioxman are up and running

*Jul 15 10:00:51.896: [IOX DEBUG] Building the verbose VPG-based enable command string

*Jul 15 10:00:51.901: [IOX DEBUG] Enable command is: request platform software iox-manager
app-hosting guestshell enable VirtualPortGroup 31 guest-ip 192.168.2.2 gateway 192.10

*Jul 15 10:00:51.902: [IOX DEBUG] Issuing guestshell enable command and waiting for it to be
up
The process for the command is not responding or is otherwise unavailable
The process for the command is not responding or is otherwise unavailable
The process for the command is not responding or is otherwise unavailable
The process for the command is not responding or is otherwise unavailable
The process for the command is not responding or is otherwise unavailable

*Jul 15 10:00:57.571: [IOX DEBUG] Waiting for another 5 secs

*Jul 15 10:03:06.900: %PLATFORM-4-ELEMENT_WARNING:  F0: smand:  RP/0: Used Memory value 89%
exceeds warning level 88%Guestshell enabled successfully
*Jul 15 10:03:34.854: [IOX DEBUG] Checking for guestshell mount path

*Jul 15 10:03:34.856: [IOX DEBUG] Validating if guestshell is ready for use

*Jul 15 10:03:34.857: [IOX DEBUG] Guestshell enabled successfully

*Jul 15 10:03:37.739: %IOSXE_INFRA-6-PROCPATH_CLIENT_HOG: IOS shim client 'sman dc 0 bipc'
has taken 1574 msec (runtime: 842 msec) to process a 'unknown' message
*Jul 15 10:03:40.830: %VXE_VNIC_IF-4-DRIVER_NOT_SUPPORTED: Ignoring interface with address
1a:41:81:a8:0f:3d using unsupported veth driver.


******ZTP is beginning!******


SerialNo parsing...


Serial number: 9KZE846DMPS


Downloading configuration from URL: http://172.16.20.5/device_config.json
```

```
Destination filename [device_config.json]?
Accessing http://172.16.20.5/device_config.json...
Loading http://172.16.20.5/device_config.json
767 bytes copied in 0.274 secs (2799 bytes/sec)

Configuration commands: Collected!


Configuring device...

hostname: ZTPDevice1
ip address: 172.16.20.100
netmask: 255.255.255.0

Line 1 SUCCESS: hostname ZTPDevice1
Line 1 SUCCESS: int loopback 1
Line 2 SUCCESS: ip add 172.16.20.100 255.255.255.0
Line 3 SUCCESS: no shut
Line 4 SUCCESS: end
Line 1 SUCCESS: line con 0
Line 2 SUCCESS: logging synchronous
Line 3 SUCCESS: no exec-timeout
Line 4 SUCCESS: end
Line 1 SUCCESS: ip domain-name domain.com
Line 1 SUCCESS: crypto key generate rsa
**CLI Line # 1: The name for the keys will be: ZTPDevice1.domain.com
**CLI Line # 1: Choose the size of the key modulus in the range of 360 to 4096 for
your
**CLI Line # 1:   General Purpose Keys. Choosing a key modulus greater than 512 may
take
**CLI Line # 1:   a few minutes.
**CLI Line # 1: How many bits in the modulus [512]: % Generating 512 bit RSA keys,
keys will be non-exportable...
**CLI Line # 1: [OK] (elapsed time was 0 seconds)

Line 1 SUCCESS: username User password admin
Line 1 SUCCESS: enable secret ztpdevice1
Line 1 SUCCESS: line vty 0 4
Line 2 SUCCESS: transport input ssh
Line 3 SUCCESS: login local


Configuration applied!


******ZTP is finished!******


ZTPDevice1#
```

Slika 20 Prikaz logova s jednog od uređaja na kojima se izvodio ZTP proces, a koji se javljaju prilikom pokretanja Cisco mrežnog uređaja i izvršavanja skripte

Nakon što se završi proces pokretanja *Guest Shell* okruženja, započinje izvođenje skripte u *Python* okruženju. Na isječku prikazanom na slici 20 moguće je vidjeti kako izgleda izvršavanje skripte i prikaz uspješno izvedenih naredbi predanih unutar funkcija *Python* CLI modula. Nakon završetka ZTP procesa uređaj je konfiguriran i spreman za obavljanje definirane funkcije na mreži.

Iz priloženog primjera primjene ZTP modela moguće je vidjeti njegove prednosti. U ovom slučaju je korišteno manje okruženje, no isti je model moguće primijeniti u mnogo većim razmjerima. Razvojem jedne skripte i konfiguracijske datoteke, u koju bilo tko može uvrstiti odgovarajuće podatke, moguće je konfigurirati velik broj uređaja prilikom čega je potrebno samo uređaj povezati na mrežu.

7. Zaključak

Današnje tvrtke i organizacije predstavljaju računalnim mrežama nove i sve strože zahtjeve, kao što su povećan fokus na jednostavnost, automatizaciju, fleksibilnost, inteligentne mehanizme za povratne informacije te svijest o aplikacijama, korisnicima i uređajima. Kao rezultat toga javlja se nova paradigma softverski definiranih mreža (SDN) koje odvajaju kontrolni dio mreže od dijela prosljeđivanja čime se stvara centralizirana programibilna infrastruktura. Tako je za širokopojasni pristup mreži razvijeno SD-*Access* rješenje koje odgovara na navedene izazove omogućujući *end-to-end* automatizaciju i sigurnost temeljenu na politici, kako za žičani, tako i za bežični pristup mreži.

Svakako jedan od najvećih izazova u tradicionalnim mrežama jest postupak implementacije novih uređaja na mrežu. Tako implementacija novih uređaja iziskuje izlazak mrežnog stručnjaka na lokaciju implementacije i ručnu konfiguraciju putem sučelja naredbenog retka što u mrežama većih razmjera predstavlja dugotrajan i troškovno zahtjevan proces. SD-*Access* tu nudi rješenje u obliku ZTP modela koji pruža priliku da se proces implementacije novih uređaja znatno pojednostavi.

U ovom radu prikazan je jedan primjer implementacije ZTP modela u emuliranoj mreži izrađenoj po uzoru na realna poslovna okruženja predstave sve funkcionalnosti istog čija bi primjena mrežnim stručnjacima znatno pojednostavila proces implementacije mrežnih uređaja u mrežu, a tvrtki tako smanjila troškove. Realizacija ZTP procesa u emuliranoj mreži potvrdila je prethodna istraživanja po pitanju efikasnosti, gdje su prethodno navedeni autori naglašavali njegove prednosti. ZTP proces proveden je na dva uređaja na različitim mrežnim lokacijama upotrebom centraliziranog poslužitelja.

Bitno je naglasiti da je prikaz ZTP modela izvršen u mreži temeljenoj na tradicionalnoj arhitekturi. Naime, koncept SD-*Access* arhitekture mreže zahtjeva još neko vrijeme da bude spreman za implementaciju u realnom okruženju, no ZTP rješenje moguće je primjenjivati već sada u postojećim mrežama. Jedina negativna strana implementacije ZTP modela jest da iziskuje upotrebu uređaja s novijim operativnim sustavom (u slučaju upotrebe *Cisco* uređaja to su IOS XE, IOS XR ili NX-OS) koji su zasada još uvijek poprilično skupi te kao takvi nisu još uvijek popularni

u realnim mrežnim implementacijama. Međutim, tijekom godina sigurno se može očekivati da će ZTP model biti zastupljeniji na različitim modelima uređaja i da će tvrtke koje se bave integracijom mreža spoznati sve prednosti primjene ovog modela što je svakako dodatna vrijednost obrađene teme ovog diplomskog rada.

Idući korak u radu na obrađenoj temi svakako bi bio provesti emulirani model u realnom fizičkom okruženju. Iako je diplomski rad opisivao primjenu ZTP modela u tradicionalnom mrežnom okruženju, pri čemu se nastojalo barem jednim dijelom automatizirati postojeće mreže, potrebno bi bilo ispitati provedbu ovog modela u okruženjima softverski definiranih mreža, pri čemu se koncept svakako može proširiti i na softverski definirano WAN okruženje.

Popis kratica i akronima

ACL	Access Control List
API	Application Programming Interface
B.Y.O.D.	Bring Your Own Device
CAPWAP	Control and Provisioning of Wireless Access Points
CLI	Command-Line Interface
CPE	Customer Premises Equipment
DHCP	Dynamic Host Configuration Protocol
DNA	Digital Network Architecture
DSL	Digital Subscriber Line
ECMP	Equal-Cost Multi-Path Routing
EID	Endpoint Identifiers
ForCES	Forwarding and Control Element Separation
FTTP	Fiber to the Premises
HAL	Hardware Abstraction Layer
HSRP	Hot Standby Router Protocol
HTTP	HyperText Transfer Protocol
IBS	Intent-Based Networking
ISE	Identity Service Engine
NOS	Network Operating System
NV-RAM	Non-Volatile Random Access Memory
OF-Config	OpenFlow Config
OVSDB	vSwitch Database Protocol
PAD	vSwitch Database Protocol
PnP	Plug-n-Play
POF	Protocol Oblivious Forwarding
RAM	Random Access Memory
REST	Representational State Transfer
RLOC	Routing Locators

ROFL	Revised OpenFlow Library
ROM	Read-OnlyMemory
SD-Access	Software-Defined Access
SDN	Software-Defined Access
SGT	Scalable Group Tag
SNMP	Scalable Group Tag
STP	Spanning Tree Protocol
TFTP	Trivial File Transfer Protocol
URL	Uniform Resource Locator
VN	Virtual Networks
VRRP	Virtual Router Redundancy Protocol
VXLAN	Virtual Extensible LAN
WAN	Wide Area Network
WLC	Wireless LAN Controller
ZTP	Zero-Touch Provisioning

Literatura

[1] Szigeti T, Zacks D, Falkner M, Simone A. Cisco Digital Network Architecture: Intent-based Networking for the Enterprise. Hoboken: Cisco Press; 2018. 753 str.

[2] Filiposka S, Demchenko Y, Arbel D, Mishev A, De Vos M, Karaliotas T, i ostali. Enabling High Performance Cloud Computing Using Zero Touch Provisioning. U: 23rd Telecommunications Forum. Beograd: IEEE; 2015. str. 67–70.

[3] Demchenko Y, Filiposka S, Tuminauskas R, Mishev A, Baumann K, Regvart D, i ostali. Enabling Automated Network Services Provisioning for Cloud Based Applications Using Zero Touch Provisioning. U: Proceedings - 2015 IEEE/ACM 8th International Conference on Utility and Cloud Computing, UCC 2015. Limassol: ACM; 2015. str. 458–64.

[4] Ashraf U. Placing controllers in software-defined wireless mesh networks. U: 2018 International Conference on Computing, Mathematics and Engineering Technologies: Invent, Innovate and Integrate for Socioeconomic Development, iCoMET 2018 - Proceedings. Sukkur: IEEE; 2018. str. 1–4.

[5] Kreutz D, Ramos FM V., Verissimo PE, Rothenberg CE, Azodolmolky S, Uhlig S. Software-Defined Networking : A Comprehensive Survey. Proc IEEE. 2015.;103(1):14–76.

[6] Neghabi AA, Navimipour NJ, Hosseinzadeh M, Rezaee A. Load Balancing Mechanisms in the Software Defined Networks: A Systematic and Comprehensive Review of the Literature. IEEE Access. 2018.;6(March):14159–78.

[7] Prajapati A, Sakadasariya A, Patel J. Software defined network: Future of networking. U: Proceedings of the 2nd International Conference on Inventive Systems and Control, ICISC 2018. Coimbatore: IEEE; 2018. str. 1351–4.

[8] Booch G. SDN architecture. Open Netw Found. 2014.;23(2):16–8.

[9] Vijay Tijare P, Vasudevan D. The Northbound APIs of Software Defined Networks. Int J Eng Sci Res Technol. 2016.;501(October). Preuzeto sa: http://www.ijesrt.com [Pristupljeno: lipanj 2020.]

[10] Han S, Lee S. Implementing SDN and network-hypervisor based programmable network using Pi stack switch. U: International Conference on ICT Convergence 2015: Innovations Toward the IoT, 5G, and Smart Media Era, ICTC 2015. Jeju: IEEE; 2015. str. 579–81.

[11] Rao S. SDN and its use-cases-NV and NFV. Semantic Scholar. 2014. str. 26. Preuzeto sa: https://in.nec.com/en_IN/pdf/NTI_whitepaper_SDN_NFV.pdf [Pristupljeno: lipanj 2020.]

[12] Bakshi K. Considerations for Software Defined Networking (SDN): Approaches and use cases. U: IEEE Aerospace Conference Proceedings. Big Sky: IEEE; 2013. str. 1–9.

[13] Kerpez K, Cioffi J, Ginis G, Goldburg M, Galli S, Silverman P. Software-defined access networks. IEEE Commun Mag. 2014.;52(9):152–9.

[14] Cisco Systems. Cisco Software-Defined Access Enabling intent-based networking 2nd edition. 2019. Preuzeto sa: https://www.cisco.com/c/dam/en/us/products/se/2018/1/Collateral/nb-06-software-defined-access-ebook-en.pdf [Pristupljeno: lipanj 2020.]

[15] Cisco Systems. Software-Defined Access Solution Design Guide. 2020. Preuzeto sa: https://www.cisco.com/c/en/us/td/docs/solutions/CVD/Campus/cisco-sda-design-guide.pdf [Pristupljeno: lipanj 2020.]

[16] Cisco Systems. Locator ID Separation Prototocol Overview. 2018. str. 1–8. Preuzeto sa: https://www.cisco.com/c/en/us/td/docs/ios-xml/ios/iproute_lisp/configuration/xe-3s/irl-xe-3s-book/irl-overview.pdf [Pristupljeno: lipanj 2020.]

[17] Cisco Systems. SD-Access Segmentation Design Guide. 2018. str. 37. Preuzeto sa: https://www.cisco.com/c/dam/en/us/td/docs/solutions/CVD/Campus/CVD-Software-Defined-Access-Segmentation-Design-Guide-2018MAY.pdf [Pristupljeno: lipanj 2020.]

[18] Hafner V. Software-Defined Access. 2018. Preuzeto sa: https://networking.combis.hr/wp-content/uploads/2018/03/ComWan_i_SD-Access_sve_sto_vasa_mreza_treba.pdf [Pristupljeno: lipanj 2020.]

[19] Cisco Systems. LAN Automation : Step-by-Step Deployment. 2020. str. 49. Preuzeto sa: https://www.cisco.com/c/en/us/td/docs/cloud-systems-management/network-automation-and-management/dna-center/tech_notes/b_dnac_sda_lan_automation_deployment.pdf

[Pristupljeno: lipanj 2020.]

[20] Cisco Systems. SNMP. 2018. str. 1–40. Preuzeto sa: https://www.cisco.com/c/en/us/td/docs/ios-xml/ios/snmp/configuration/xe-16/snmp-xe-16-book/nm-snmp-cfg-snmp-support.pdf [Pristupljeno: lipanj 2020.]

[21] Cisco Systems. Cisco Identity Services Engine. 2019. str. 8. Preuzeto sa: https://www.cisco.com/c/en/us/products/collateral/security/identity-services-engine/data_sheet_c78-656174.pdf [Pristupljeno: lipanj 2020.]

[22] Network Direction. Hierarchy Design – Part 1. 2018. Preuzeto sa: https://networkdirection.net/articles/network-theory/hierarchicalnetworkmodel/hierarchydesignpart1/ [Pristupljeno: lipanj 2020.]

[23] Mishra R, Gijare V, Malik S. Zero Touch Network: A Comprehensive Network Design Approach. Int J Eng Res Technol. 2019.;8(09):792–4. Preuzeto sa: https://www.ijert.org/research/zero-touch-network-a-comprehensive-network-design-approach-IJERTV8IS090259.pdf [Pristupljeno: lipanj 2020.]

[24] Nass R. Zero-Touch Provisioning for 5G Networks. 2019. Preuzeto sa: https://www.insight.tech/content/zero-touch-provisioning-for-5g-networks [Pristupljeno: lipanj 2020.]

[25] Cisco DevNet. Zero-Touch Provisioning Zero-Touch. 2020. str. 1–7. Preuzeto sa: https://developer.cisco.com/docs/ios-xe/#!zero-touch-provisioning/zero-touch-provisioning [Pristupljeno: lipanj 2020.]

[26] Juniper Networks. Software Installation and Upgrade Guide. 2020. str. 898. Preuzeto sa: https://www.juniper.net/documentation/en_US/junos/information-products/pathway-pages/software-installation-and-upgrade/software-installation-and-upgrade.pdf [Pristupljeno: lipanj 2020.]

[27] Juniper Networks. Understanding VXLANs. 2019. Preuzeto sa: https://www.juniper.net/documentation/en_US/junos/topics/topic-map/sdn-vxlan.html [Pristupljeno: lipanj 2020.]

[28] Arista Networks. Zero Touch Provisioning. 2016. Preuzeto sa:

https://www.cisco.com/c/en/us/td/docs/switches/metro/me1200/controller/guide/b_nid_controller_book/b_nid_controller_book_chapter_011.pdf [Pristupljeno: lipanj 2020.]

[29] Weinberg N. The 10 most powerful companies in enterprise networking. Network World. 2019. Preuzeto sa: https://www.networkworld.com/article/3211410/the-10-most-powerful-companies-in-enterprise-networking.html [Pristupljeno: lipanj 2020.]

[30] Cisco Systems. Guest Shell. 2020. str. 14. Preuzeto sa: https://www.cisco.com/c/en/us/td/docs/ios-xml/ios/prog/configuration/166/b_166_programmability_cg/guest_shell.pdf [Pristupljeno: lipanj 2020.]

[31] Byrne B. Introduction to GuestShell. 2018. Preuzeto sa: https://www.ciscolive.com/c/dam/r/ciscolive/us/docs/2018/pdf/DEVNET-1695.pdf [Pristupljeno: lipanj 2020.]

[32] Cisco Systems. Campus LAN and Wireless LAN Design Guide. 2020. Preuzeto sa: https://www.cisco.com/c/en/us/td/docs/solutions/CVD/Campus/cisco-campus-lan-wlan-design-guide.pdf [Pristupljeno: lipanj 2020.]

[33] Juniper Network. Network Configuration Example Midsize Enterprise Campus Solution Configuration Example. 2019. str. 178. Preuzeto sa: https://www.juniper.net/documentation/en_US/release-independent/nce/information-products/pathway-pages/nce/nce-143-midsize-campus.pdf [Pristupljeno: lipanj 2020.]

Popis slika

Slika 1 Razlika između tradicionalne i SDN arhitekture 5
Slika 2 Softverski definirane mreže prikazane kroz: a) razine, b) slojeve i c) arhitekturu dizajna sustava 7
Slika 3 Prikaz dijagrama namjenski temeljene mreže za digitalno poslovanje 13
Slika 4 Prikaz overlay i underlay slojeva 15
Slika 5 Virtualne mreže i skalabilne grupe u SD-Access mreži 18
Slika 6 Prikaz primjera proširene podmreže 19
Slika 7 Prikaz granice automatizacije koju podržava Cisco DNA Centar 23
Slika 8 SD Access rješenje i prikaz elemenata fabric topologije. 26
Slika 9 Prikaz SD-Access implementacija bežične mreže. 33
Slika 10 Pikaz Over-the-top implementacija bežične mreže 34
Slika 11 Dijagram toka tradicionalnog načina implementacije mrežnog uređaja 36
Slika 12 Izvođenje ZTP procesa po koracima 37
Slika 13 Pregled IOx platforme 39
Slika 14 Snimka zaslona radne okoline softvera EVE-NG koja prikazuje topologiju mreže korištene za testiranje ZTP procesa 43
Slika 15 Snimka zaslona radne okoline softvera EVE-NG koja prikazuje topologiju mreže sa konfiguriranim mrežnim elementima, pridruženim IP adresama i definiranim VLAN-ovima 45
Slika 16 Prikaz konfiguracije DHCP poslužitelja na Linux poslužitelju 46
Slika 17 Prikaz konfiguracije DHCP poslužitelja na Cisco uređaju 47
Slika 18 Python skripta namijenjena za izvođenje na IOS XE Cisco uređaju 49
Slika 19 Prikaz konfiguracijske datoteke u JSON formatu 50
Slika 20 Prikaz logova s jednog od uređaja na kojima se izvodio ZTP proces, a koji se javljaju prilikom pokretanja Cisco mrežnog uređaja i izvršavanja skripte 54

Sveučilište u Zagrebu
Fakultet prometnih znanosti
10000 Zagreb
Vukelićeva 4

IZJAVA O AKADEMSKOJ ČESTITOSTI I SUGLASNOST

Izjavljujem i svojim potpisom potvrđujem kako je ovaj diplomski rad isključivo rezultat mog vlastitog rada koji se temelji na mojim istraživanjima i oslanja se na objavljenu literaturu što pokazuju korištene bilješke i bibliografija.

Izjavljujem kako nijedan dio rada nije napisan na nedozvoljen način, niti je prepisan iz necitiranog rada, te nijedan dio rada ne krši bilo čija autorska prava.

Izjavljujem također, kako nijedan dio rada nije iskorišten za bilo koji drugi rad u bilo kojoj drugoj visokoškolskoj, znanstvenoj ili obrazovnoj ustanovi.

Svojim potpisom potvrđujem i dajem suglasnost za javnu objavu diplomskog rada pod naslovom **Automatizacija procesa konfiguracije preklopnika uporabom modela Zero-Touch Provisioning** na internetskim stranicama i repozitoriju Fakulteta prometnih znanosti, Digitalnom akademskom repozitoriju (DAR) pri Nacionalnoj i sveučilišnoj knjižnici u Zagrebu.

Student/ica:

U Zagrebu, 16.9.2020

Ivan Šimunić I. Šimunić
(potpis)