VAASAN AMMATTIKORKEAKOULU
UNIVERSITY OF APPLIED SCIENCES

Helene Hujic

VERKKOJEN OHJELMOITAVUUS

Tekniikka
2020

VAASAN AMMATTIKORKEAKOULU
Tietotekniikka

TIIVISTELMÄ

Tekijä	Helene Hujic
Opinnäytetyön nimi	Verkkojen ohjelmoitavuus
Vuosi	2020
Kieli	suomi
Sivumäärä	34 + 2 liitettä
Ohjaaja	Antti Virtanen

Tämän opinnäytetyön aikana tutustutaan tietoverkkojen laitteiden ohjelmoitavuuteen. Tarkoituksena oli luoda monen laitteen muokkaukseen soveltuva ratkaisu ja tutustua Postman-alustan avulla YANG-moduulien mukaiseen laitekonfigurointiin.

Teoriaosuudessa tutustutaan keskeisimpiin verkkojen ohjelmoitavuuteen liittyviin käsitteisiin ja niiden käyttötarkoituksiin. Opinnäytetyön aikana käytetään Python-ohjelmointikieltä luomaan toimiva ratkaisu, sekä Postman-alustaa käsittelemään YANG-mallin mukaisia moduuleja yhdessä RESTCONF-protokollan kanssa.

Opinnäytetyön tuloksena on Python-ohjelmointikielellä luotu ohjelmakoodi, jolla voidaan muokata useamman laitteen konfiguraatioita. Lisäksi käytetään Postman-alustaa muokkaamaan IPv6-osoitteita, sekä luomaan VLAN-rajapinta laitteelle. Näiden muutosten aikaansaaminen vaatii YANG-mallin Cisco-IOS-XE-native-moduulin käyttöä.

Avainsanat ohjelmoitavuus, YANG, RESTCONF, Python

VAASAN AMMATTIKORKEAKOULU
UNIVERSITY OF APPLIED SCIENCES
Information Technology

ABSTRACT

Author	Helene Hujic
Title	Network Programmability
Year	2020
Language	Finnish
Pages	34 + 2 appendices
Name of Supervisor	Antti Virtanen

During this thesis, you will familiarize yourself with the programmability of computer network equipment. The aim was to create a solution suitable for customizing many devices and to use the Postman platform to familiarize yourself with the YANG modules.

The theory section explores the main concepts related to the programmability of networks and their uses. During the thesis, the Python programming language is used to create a functional solution, as well as the Postman platform to handle modules based on the YANG model together with RESTCONF.

During the thesis, a program code is created in the Python programming language to modify the configurations of multiple devices. In addition, the Postman platform is used to modify IPv6 addresses and to create a VLAN interface for the device. To make these changes, you need to use the Cisco-IOS-XE-native module of the YANG model.

Keywords Programmability, YANG, RESTCONF and Python

SISÄLLYS

KUVALUETTELO

KÄSITTEET JA LYHENTEET

AAA	Authentication, Authorization and Accounting, Autentikointi, valtuutus ja tilastointi, protokolla, jolla voidaan tunnistaa ja todentaa käyttäjä verkossa.
API	Application Programming Interface, Ohjelmointirajapinta
ASIC	Application-Specific Integrated Circuit, sovelluskohtainen mikropiiri
CLI	Command Line Interface, Komentokehote
EIGPR	Enhanced Interior Gateway Routing Protocol, Reititysprotokolla.
FDN	Finger Defined Networking, verkkojen toteuttaminen manuaalisesti
HTTP	Hypertext Transfer Protocol, Tiedonsiirtoprotokolla.
IETF	Internet Engineering Task Force, organisaatio, joka vastaa Internet-protokollien standardoinnista
IP	Internet Protocol, TCP/IP-mallin verkkokerrosprotokolla
IPv6	Internet Protocol version 6, verkkokerrosprotokollan 128-bittiset osoitteet
JSON	JavaScript Object Notation, sekä tiedostomuoto että formaatti tiedonvälitykseen
LAN	Local Area Network, lähiverkko
NETCONF	Network Configuration protocol, verkon määritysprotokolla
OSPF	Open Shortest Path First, reititysprotokolla

REST	REpresentational State Transfer, HTTP-protokollaan perustuva arkkitehtuuri.
RESTCONF	REST like Configuration Protocol, verkon määritysprotokolla RESTin avulla
RFC	Request for Comments, IEFT-organisaation julkaisemia standardeja koskien Internetiä
RPC	Remote Procedure Call, etäproseduurikutsu, käytetään hajautettujen järjestelmien toteuttamiseen.
SDN	Software Defined Networking, verkkojen toteuttaminen ohjelmallisesti
SSH	Secure Shell, protokolla, jolla voidaan salata tietoliikenne
VLAN	Virtual LAN, virtuaalinen lähiverkko
XML	Extensible Markup Language, sekä tiedostomuoto että formaatti tiedonvälitykseen
YANG	Yet Another Next Generation, datan mallinnuskieli

1 JOHDANTO

Tietoliikenneverkkojen laitemäärien kasvaessa tarve automatisoinnille kasvaa. Jokainen laite täytyy konfiguroida tiettyyn konfiguraatioon, jotta se täyttäisi organisaation vaatimukset ja nykyiset vaatimukset tietoturvalle. Perinteisesti nämä konfiguraatiot on tehty yksitellen jokaiseen laitteeseen käsin joko terminaaliyhteyden tai terminaaliemulaattorin, kuten PuTTYn avulla. Tämä kuitenkin aiheuttaa sen, että jos organisaation laitteisiin täytyy tehdä muutoksia, perinteisellä menetelmällä se on hyvin aikaa vievää ja hidasta. Automatisoinnin tarve on suuri varsinkin suurissa organisaatioissa. Verkkojen ohjelmoitavuus on vastaus tähän automatisoinnin tarpeeseen. Käyttämällä joko ohjelmakoodeja tai jo valmiita SDN-ratkaisuja, voidaan verkkojen muokkaus ja valvonta tehdä ilman, että järjestelmänvalvojan täytyy ottaa terminaaliyhteys laitteeseen. Laite voi myös fyysisesti olla eri tilassa kuin järjestelmänvalvoja.

Tässä opinnäytetyössä tutustutaan verkkojen ohjelmoitavuuteen liittyviin käsitteisiin ja protokolliin. Näistä tärkeimpinä YANG, RESTCONF ja NETCONF. Näihin käsitteisiin tutustutaan ensin harjoitteissa, joiden jälkeen luodaan ohjelmakoodiratkaisu, jolla voidaan muokata useamman laitteen konfiguraatioita. Lisäksi tutustaan konfiguraatioiden muokkaukseen Postman-alustan avulla. Alustalla luodaan VLAN-rajapinta, aktivoidaan IPv6-osoitteiden käyttö ja annetaan rajapinnalle IPv6-osoite.

2 TAUSTA, TARKOITUS JA TAVOITTEET

Ideana tässä opinnäytetyössä on ollut tutustua, kuinka tietoliikenneverkkoa voitaisiin hallinnoida ohjelmoinnin avulla. Tulevaisuudessa tietoverkkojen koko tulee kasvamaan ja uusille laitteille tulee kysyntää koko ajan. Tämä kasvava määrä vaatii tietoverkkojen ylläpitäjiltä kasvavassa määrin tehokkuutta ja jokaisen laitteen yksi kerrallaan konfigurointi veisi liikaa aikaa. Network Programmability tuo tähän tarpeeseen vastauksen automatisoimalla verkon valvonnan, konfiguroinnin ja ongelman etsinnän.

Nykyiseltään verkon ylläpitäjät konfiguroivat laitteet FDN-periaatteella. FDN eli Finger Defined Networking tarkoittaa, että laitteet konfiguroidaan yksitellen, syöttämällä komennot suoraan laitteeseen, käyttämällä CLItä (Command Line Interface). Tämä on kuitenkin aikaa vievää ja vaatii manuaalista konfiguraatioiden varmistusta laite kerrallaan. Tämän manuaalisuuden takia virheiden määrä kasvaa, sillä jokaisen laitteen konfigurointi vaatii useampaa painallusta hiirellä. Lisäksi suurin osa laitteista saa samanlaisen konfiguraation. Esim. saman yrityksen saman haaran lähes kaikki kytkimet ohjataan välittämään dataliikenne samalle reitittimelle käyttämällä samaa oletusyhdyskäytävää. Tämä asetetaan kaikkiin Cisco kytkimiin samaa komentoa käyttäen (IP default-gateway 0.0.0.0). Vaikka tämä komento ei ole kovinkaan pitkä se vaatii kuitenkin tarkkaavaisuutta, jotta jokaisella kytkimellä olisi sama oletusyhdyskäytävän IP-osoite. Käyttämällä SDN:ää eli Software Defined Networking, voidaan tämä kyseinen komento välittää ohjelmakoodin avulla /19/.

SDN:n avulla verkon ylläpitäjä pystyy tekemään laitekonfiguraatiot muutamalla hiiren painalluksella. Jokaiselle laitteelle voidaan lähettää identtiset yleiskonfiguraatiot ja näin tehtäessä virheiden määrä laskee. Lisäksi samaan ohjelmakoodiin voidaan tehdä laitekohtaisia muutoksia, jotta jokaiselle laitteelle saadaan uniikit laitenimet, IP-osoitteet ja tarvittaessa virtuaaliverkot, joilla voidaan erotella esimerkiksi työntekijät vierailijoista.

Tässä opinnäytetyössä on tarkoituksena tutustua kuinka käytännössä Network Programmability toimii. Opinnäytetyön aikana on ensin tutustuttu Cisco Academyn

tuottamaan Network Programmability materiaaliin ja harjoitteisiin. Tarkoituksena on ollut nähdä, kuinka käytännössä tietoverkon laiteita voidaan ohjelmoida. Lisäksi on ollut tarkoituksena tutustua, kuinka ohjelmointikurssien aikana tutuksi tulleita käsitteitä kuten REST ja JSON/XML voidaan soveltaa tietoverkkolaitteiden konfiguroinnissa.

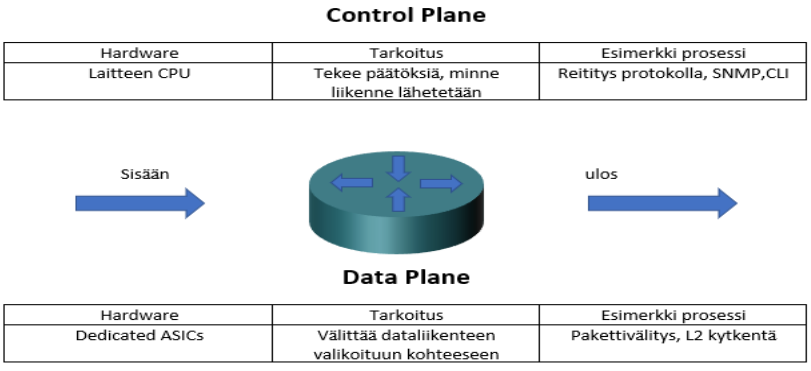
3 TEORIATAUSTA

Network Programmability eli tietoverkon ohjelmoinnin toteutus teoriassa tarkoittaa tietoverkon konfigurointia ohjelmoinnin avulla. Kaikki tarvittavat komennot täytyy välittää reitittimiin ja kytkimiin oikean muotoisina, jotta laitteet tunnistaisivat nämä annetut komennot ja toimisivat halutulla lailla. Tämä tarkoittaa, että käytössä olevien laitteiden tulee tukea tätä tekniikkaa, jotta toteutus on mahdollista. Komentojen välitykseen tarvitaan REST-rajapintoja, joilla voidaan ohjata PUT-, GET-, POST- ja DELETE-komentojen avulla datan suuntaa. YANG-mallin avulla muotoillaan komennot JSON/XML-muodoista laitteiden ymmärrettävään muotoon.

3.1 Tuetut laitteet

Cisco Systemsin tuotteet, jotka tukevat Cisco IOS XE 16-ohjelmistoa, tukevat natiivisti ohjelmoitavuutta. Nämä laitteet tukevat myös muita teknologioita, joita tässä opinnäytetyössä sivutaan /1/. Näistä laitteista, varsinkin reitittimistä puhuttaessa, täytyy myös mainita kuinka nämä laitteet ohjaavat informaatiota.

Informaation ohjaamiseen reitittimet ja kytkimet käyttävät käsitteellistä mallia nimeltä taso (plane). Näitä tasoja on kaksi: datataso (data plane) ja ohjaustaso (control plane), jota kutsutaan myös hallintatasoksi. Kuvassa 1 on kuvattuna tasojen eroavaisuus.



Kuva 1. Control Plane & Data Plane /25/

Laitteiden päätöksenteko tehdään ohjaustasolla, joka on ohjelmistopohjainen. Tämä taso käyttää prosessorin resursseja, kun taas datataso käyttää erikoistuneempia laitteistoja kuten ASIC (Application-Specific Integrated Circuit). Tietoliikenne, joka on joko laitteisiin lähettyä tai laitteen generoimaa, on ohjaustason liikennettä. Ohjaustason liikennettä on myös esim. AAA:han kuuluva liikenne. AAA eli Authentication, Authorization and Accounting on verkon hallintaa, jonka avulla voidaan seurata käyttäjien aktiviteetteja tietoverkossa, hallita heidän pääsyään tiettyihin aineistoihin sisäverkossa ja varmentaa käyttäjien identiteetin /12/. Ohjaustasolta ohjataan myös resursseja datatasolle. Tätä ohjausta tapahtuu esim., kun reititin muodostaa naapurisuhteita OSPF:n (Open Shortest Path First) tai EIGPR:n (Enhanced Interior Gateway Routing Protocol) aikana. Ohjaustasoa voidaan myös kutsua hallintatasoksi, sillä sitä voidaan käyttää laitteiden hallintaan esim. SSH:n avulla. Laitteiden datatason liikenne on pakettien ohjausta eteenpäin. Tämän liikenteen luonteen takia, tasoa kutsutaankin uudelleenohjaustasoksi. Tämän täytyy olla tarkkaa, jotta pakettien sisältö pääsisi oikeaan kohdelaitteeseen, ja nopeaa, jotta paketit välittyisivät mahdollisimman pienellä viiveellä /2/. Laitteiden hallintataso voi olla myös SDN-määritelty.

3.2 SDN

SDN eli Software Defined Networking on verkkoarkkitehtuuri, jolla voidaan virtualisoida verkko. Hallintatasolla tarkoitetaan, että se olisi korvattu SDN-ohjaimella. Tämä ohjain keskittäisi hallinnan ja voisi hallita useampaa datatasoa saman aikaisesti. Ilman SDN-ohjainta, jokaisella datatasolla olisi oma hallintataso /10/.

SDN-arkkitehtuurin avulla voidaan hallita ja monitoroida verkkoa. Sen komponentteina voidaan käyttää Openflow:ta ja OpenStackia. Openflow on kehitetty Stanfordin yliopistolla, ja sen tehtävänä on hallinnoida liikennettä eri laitteiden ja ohjaimen välillä. OpenStack on alusta, jonka avulla voidaan virtualisoida ja orkestroida erilaisia esim. skaalattavia pilviympäristöjä /11/.

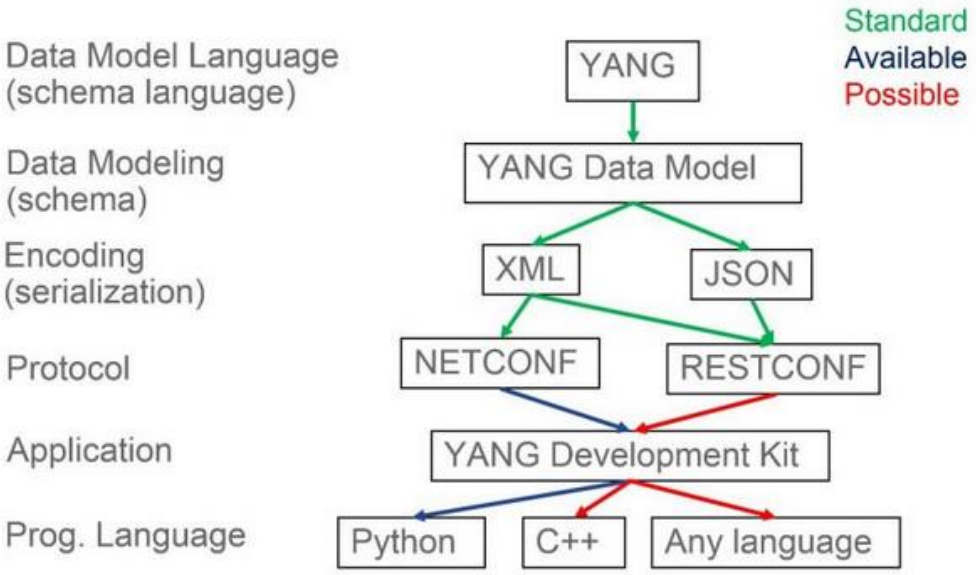
3.3 RESTful API

REST eli REpresentational State Transfer on HTTP-protokollaan pohjautuva arkkitehtuurimalli ja sillä toteutetaan ohjelmointirajapintoja. RESTin avulla voidaan muokata tilaa sekä asiakkaan että palvelimen päässä. Arkkitehtuurin REpresentational sanalla viitataan käsittelyssä olevan datan eli resurssin muotoon. Tätä dataa muokataan siten, että sen esitysmuodoksi tulee esim. XML (Extensible Markup Language). State viittaa resurssin tilaan sekä asiakassovelluksella että palvelimella. REST-kutsujen tarkoitus on muokata tai lukea tätä tilaa. Tämän tilan muuttamiseen käytetään HTTP:n tuomia GET-, POST-, PUT-, ja DELETE-metodeja. RESTin Transfer sanalla viitataan juuri tähän tilan muokkaamiseen ja tarkasteluun tehtyjä kutsuja /3/.

REST-arkkitehtuuria kutsutaan yleisesti myös sanalla RESTful, mutta sen täytyy täyttää tietyt vaatimukset, jotta sitä voitaisiin kutsua tällä nimellä. Näitä vaatimuksia on kuusi: Client-server, Stateless, Cache, Uniform interface, Layered system ja Code on demand (valinnainen). Client-server tarkoittaa, että järjestelmän on sisällettävä palvelimia ja asiakaspäätteitä. Palvelimet nimensä mukaisesti palvelevat asiakaspäätteitä esimerkiksi tarjoamalla tietokannan, joita asiakaspääte käyttää toteuttaakseen käyttöliittymiä. Stateless sanalla viitataan palvelin-asiakaspäätteiden kommunikointiin. Tämä tarkoittaa sitä, että palvelin ei varastoi tietoa asiakaspääteistä (esim. evästeet) vaan nämä tiedot jäävät asiakaspäätteelle. Cache viittaa myös palvelin-asiakaspäätteiden kommunikointiin. Tämä tarkoittaa palvelimen päässä sitä, että joko palvelin on cacheable- tai non-cacheable-tilassa. Tämä tila vaikuttaa palvelimen vastausnopeuteen. Uniform interface viittaa siihen, että esimerkiksi samaa selainta voidaan käyttää kaikkien verkkosivujen selaukseen ilman lisäosia. Viides vaatimus Layered System, viittaa siihen, asiakaspäätteen ei tarvitse tietää keskusteleeko se suoraa palvelimen kanssa vai onko välissä jokin välityspalvelin. Asiakaspalvelimen ei tule ”nähdä” omaa kerrostaa edemmäksi, joka lisää tietoturvaa. Code on Demand on valinnainen vaatimus ja kaikista kuudesta se on ainoa, joka voi olla valinnainen. Se viittaa siihen, että esimerkiksi verkkosivuilla voi olla erilaisia ominaisuuksia, jotka asiakas/käyttäjä voi halutessaan käyttää /13/.

3.4 YANG-malli

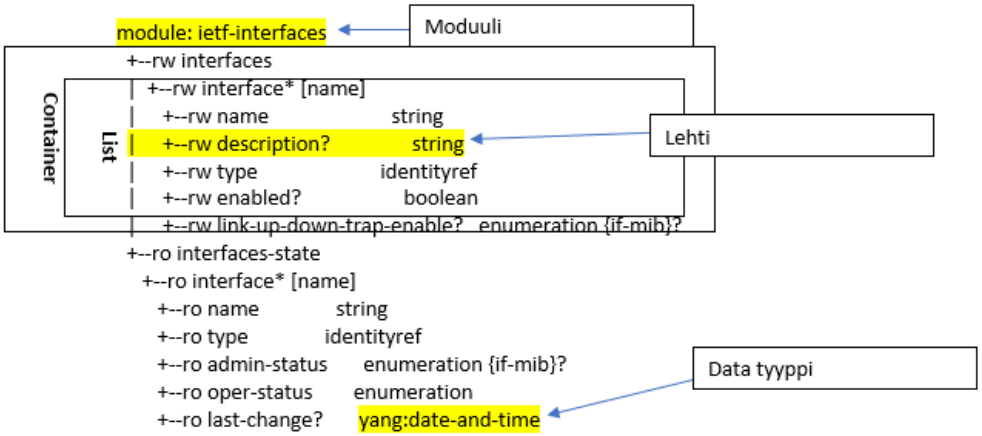
YANG on datan esitykseen käytetty kieli. Se tulee sanoista Yet Another Next Generation, ja sitä käytetään datan lähettämisessä verkonhallintaprotokollien NETCONF ja RESTCONF kanssa. Kuvasta 2 nähdään kuinka YANG liittyy NETCONF- ja RESTCONF-protokolliin.



Kuva 2. YANGin malli /14/

Reitittimille ja kytkimille lähetettyjen komentojen, kuten laitenimien asettaminen, tulee olla tietyssä muodossa, jotta laitteet ymmärtäisivät mitä komennoilla halutaan muokata. YANG on kehitetty toimimaan NETCONFin kanssa. YANG-malli muotoilee datan puumaiseksi /4/.

Kuvasta 3 voidaan nähdä YANG-datan rakenne.



Kuva 3. YANG rakenne /27/

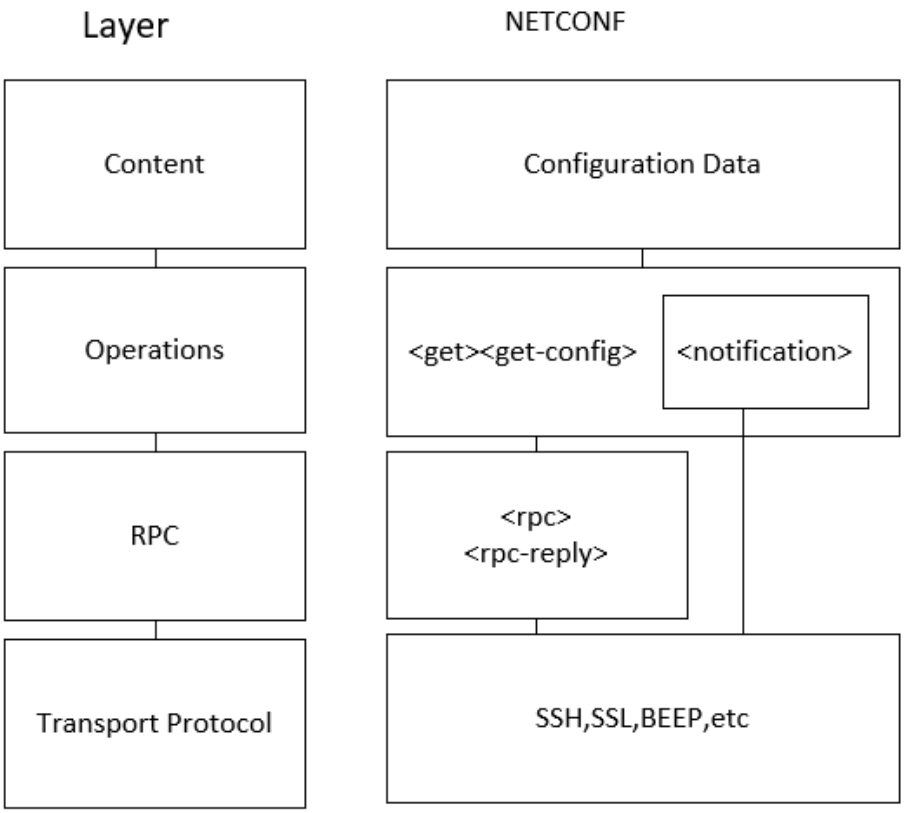
Kuvasta 3 nähdään, kuinka rakenteessa ylimpänä on moduulin nimi. Moduulin sisällä on yksi tai useampia säiliöitä (container). Säiliön sisällä on lista (leaf-list), jonka osia kutsutaan lehdiksi (leaf), ja jokaisella lehdellä on datatyyppi /4/.

3.5 NETCONF ja RESTCONF

NETCONF ja RESTCONF ovat molemmat standardoituja verkonhallintaprotokollia. Network Configuration Protocol eli NETCONF on IETF:n standardoima RFC 6241 mukainen verkonhallintaprotokolla. Sen avulla voidaan hallinta verkon laitteiden eri kokoonpanoja, ja verkon konfigurointi- ja käyttötietoja.

NETCONF-protokolla käyttää paradigmanaan RPCs (Remote Procedure Calls). /5/ Näiden kutsujen tehtävänä on välittää viestejä verkkolaitteen/-laitteiden ja verkon valvojan välillä. Kutsut koodataan XML-muotoon sekä asiakaspäätteen että palvelimen päässä./6/

NETCONF-protokollaa voidaan kuvata kerrosmallilla, kuten kuvasta 4 voidaan nähdä.



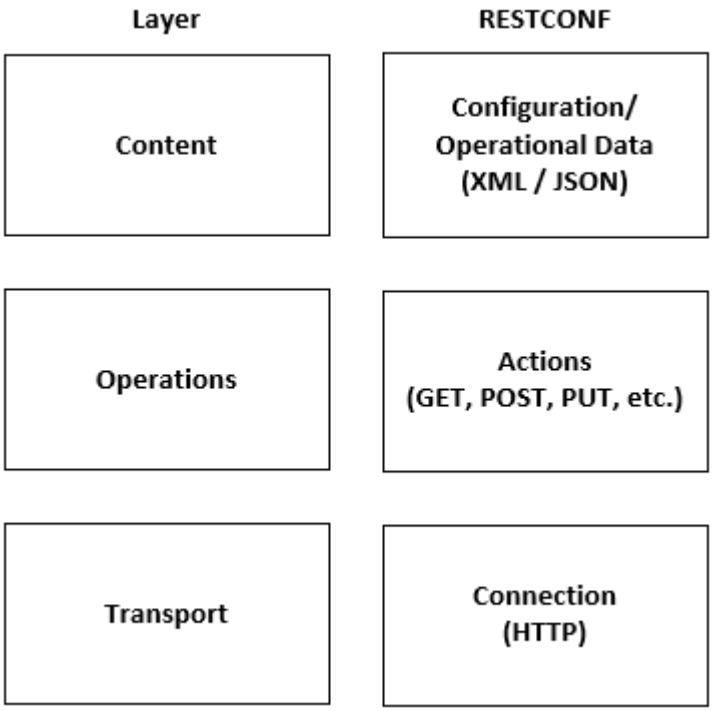
Kuva 4. NETCONFin kerrosmalli /26/

Kuvassa vasemmalla puolella on kerroksen nimi ja oikealla on mitä kerros pitää sisällään. Content viittaa YANGin mukaan konfiguroituun dataan, joka Operations tasolla muutetaan XML-muotoon. RPC viittaa RPC-kutsuihin. Transport protocol viittaa tapaan, jolla edellisen kerroksen kutsut välitetään, kuten SSH /15/.

RESTCONF on HTTP-protokollaan pohjautuva protokolla. Sen avulla voidaan konfiguroida dataa, joka on YANGissa määritelty, ja se käyttää NETCONF-protokollassa määriteltyjä datavarastokäsitteitä /7/. RESTCONF on REST-APIn kaltainen mikä tarkoittaa, että se tukee GET-, POST-, PUT-, ja DELETE-metodeja.

Kuten NETCONFissa data voi olla XML-muodossa, mutta se tukee myös JSON-muotoista dataa /8/.

Kuvassa 5 nähdään samankaltainen kerrosmalli kuten NETCONFilla.



Kuva 5. RESTCONFin kerrosmalli /28/

Ylimpänä nähdään Content, joka voi olla joko operatiivista tai konfiguraatiodataa. Tämä data voi olla joko XML- tai JSON-muotoista. Tämä data voidaan välittää Operations tasolla esim. GET-komennolla, jonka Transport taso välittää HTTP-protokollan avulla /16/.

3.6 Network Programmability

Network Programmability on Cisco Systems -yrityksen lanseeraama termi, joka tarkoittaa tietoverkon kontrollointiin käytettyä ohjelmointia. Ideana on, että verkon hallintaan voitaisiin käyttää ohjelmointia. Tämä tarkoittaa, että verkon konfigurointi, ongelman etsintä ja monitorointi tapahtuisi ilman perinteistä terminaaliyhteyttä fyysiseen laitteeseen. Ohjelmointikielenä voidaan käyttää Python-ohjelmointikieltä, mutta yksinkertaisimmillaan ohjelmointi voisi olla pelkkien REST-API pyyntöjen lähettämistä verkkolaitteelle /9/.

Työkaluina tietoverkkojen ohjelmoinnissa voidaan käyttää esim. Postman-ohjelmistoa. Ohjelmiston avulla voidaan luoda ja suorittaa esim. REST-pyyntöjä. Sitä voidaan myös käyttää testauksessa, ohjelmistojen kehityksessä ja moneen muuhun tarkoitukseen /17/. Toisena työkaluna voidaan käyttää mm. Pythonin mukana tulevaa IDLE-ohjelmistoa. Se on tehty Python-ohjelmointikielellä ja tarkoitettu käytettäväksi sen kanssa. Ympäristönä IDLE on erittäin yksikertainen. Sen avulla voidaan tehdä ohjelmakoodia ja suorittaa sitä /18/. Käytettäessä Pythonia tarvitaan avuksi kirjastoja. Tämän opinnäytetyön aikana on käytetty seuraavia kirjastoja: ncclient ja netmiko. Näistä ensimmäinen ncclient on tarkoitettu käsittelemään NETCONF-protokollaa. Sen avulla voidaan kehittää applikaatioita ja kirjoittaa ohjelmakoodia /23/. Toinen käytetty kirjasto, netmiko, jonka avulla voidaan muodostaa SSH-yhteyksiä etäyhteyden päässä olevaan laitteeseen. Se pohjautuu toiseen kirjastoon nimeltä paramiko, mutta on sen yksinkertaistetumpi muoto /20/.

Network Programmabilityn avulla voidaan automatisoida yritysten tietoverkkojen ylläpito ja se nopeuttaa uusien laitteiden lisäystä, sillä tarvittavat asetukset uusille laitteille voidaan ajaa ohjelmakoodilla tietoverkkojen ylläpitäjän omalta työpisteeltä. Tämä lisää myös tehokkuutta, sillä ylläpitäjän ei tarvitse enää tehdä asetuksia jokaiselle laitteelle yksi kerrallaan. Tämä vähentää myös virheiden määrää, sillä ylläpitäjä pystyisi tarkistamaan useamman laitteen konfiguraatiot nopeammin /19/.

4 TUTUSTUMINEN VERKON OHJELMOITAVUUTEEN

Opinnäytetyön aluksi perehdyttiin Cisco Academyn materiaaliin ja harjoituksiin Network Programmability -työpajan materiaaleilla. Nämä materiaalit antoivat alkusysäystä varsinaisen työn suunnittelua varten. Materiaalien ja harjoitteiden jälkeen tutkittiin mahdollisuutta toteuttaa jokin esim. Cisco CCNA-kurssin laboratoriotöistä niillä menetelmillä, joita harjoituksissa sivuttiin. Työpajan mukana tulee valmiiksi konfiguroitu reititintä emuloiva virtuaalikone. Jos työpaja periaatteita halutaan soveltaa fyysiselle laitteelle, tulee laite tätä varten konfiguroida aluksi käsin, jotta yhteydenotto onnistuisi. Laitteelle annetaan FDN-periaatteella SSH-yhteyttä varten salasana ja käyttäjätunnus, joita käytetään funktiossa.

4.1 Python-ohjelmointikielen käyttö verkon ohjelmoinnissa

Cisco Academyn toteuttamissa harjoituksissa tutustuttiin Python-ohjelmointikieleen ja sen mukana tulevaa IDLE-alustaa. Kaikki työpajan harjoitteet on tehty virtuaalikoneen avulla. Tällä alustalla tarvittiin myös erilaisia moduuleita. Ensimmäinen näistä käytetyistä moduuleista oli nimeltään Netmiko. Tämä moduuli yksinkertaistaa SSH CLI -yhteyden verkkolaitteisiin /20/.

Kuvassa 6 on yksi Netmiko:n mukana tulevista funktioista.

```
from netmiko import ConnectHandler

sshCli = ConnectHandler(
        device_type='cisco_ios',
        host='192.168.56.101',
        port=22,
        username='cisco',
        password='cisco123!'
        )
```

Kuva 6. Python Netmiko ConnectionHandler()-funktio

Kuvassa 6 näkyvän funktion avulla voidaan tehdä etäyhteys SSH:n avulla verkkolaitteeseen. Tarvittavina parametreina tälle funktiolle täytyy osoittaa device_type, host, port, username ja username. Device_type viittaa etäyhteyden päässä olevaan verkkolaitteeseen, joka tässä tapauksessa on Cisco IOS-pohjainen. Host on

verkkolaitteen IP-osoite ja port on SSH:n mukainen oletusportti. Username ja password ovat verkkolaitteelle asetetut SSH-yhteyden käyttäjätunnus ja salasana, jotka ovat asetettu verkkolaitteelle. Tämän lisäksi saattaa tarvita myös parametria secret, joka on Cisco-laitteiden EXEC-tilan salasana. Kuvan sshCli on käyttäjämääriteltävissä oleva objekti, jolle palautusarvoksi on asetettu ConnectionHandler()-funktio. Kaikki edellä oleva aukaisee yhteyden verkkolaitteelle, jonka jälkeen voidaan antaa konfiguroinnissa tarvittavat parametrit /20/.

Laitteen konfigurointiin käytettään seuraavan kuvan 7 listaobjektia.

```
config_commands = [
    'int loopback 1',
    'ip address 2.2.2.2 255.255.255.0',
    'description WHATEVER'
]
```

Kuva 7. Loopback rajapinnan luonti

Kuvassa 7 luodaan Loopback-rajapinta CLI-komennoilla. Config_commands on listaobjekti, johon on syötetty useammalle riville laitteelle annettavia komentoja. Tässä tapauksessa laite luo loopback 1 nimisen rajapinnan, antaa sille IP-osoitteen ja maskin, sekä antaa sille kuvauksen. Netmiko-kirjasto antaa valmiita funktioita, joilla voidaan asettaa laitteen konfiguraatiot ja antaa laitteelle eri komentoja. Tämä on havainnollistettu kuvassa 8.

```
output = sshCli.send_config_set(config_commands)

print("sending config to the device:\n{}\n".format(output))
output = sshCli.send_command("show run")
print("running config:\n{}\n".format(output))
```

Kuva 8. Netmikon mukaiset funktiot, laitteen kutsuun ja muokkaukseen

Kuvassa 8 asetetaan ensin output-muuttujalle funktio send_config_set, joka hakee sshCli-kutsulla SSH-yhteyden tiedot, ja lähettää sitten listan komennot SSH-kanavaa myöten laitteelle. Print-komento tulostaa laitteen vastauksen, jossa näkyy CLI-muotoinen vastaus, joka näkyy kuvassa 9.

```
sending config to the device:
configure terminal
Enter configuration commands, one per line.  End with CNTL/Z.
CSR1kv(config)#int loopback 1
CSR1kv(config-if)#ip address 2.2.2.2 255.255.255.0
CSR1kv(config-if)#description WHATEVER
CSR1kv(config-if)#end
CSR1kv#
```

Kuva 9. Laitteen vastaus send_config_set-funktioon

Kuvassa 9 näkyy kuinka ohjelma luo laitteelle rajapinnan automaattisesti ilman, että käyttäjän täytyisi itse kirjoittaa CLI-komennot. Send_config_set-funktio laittaa laitteen automaattisesti configure terminal -tilaan ja kirjautuu automaattisesti tilasta ulos /28/. Toinen kuvassa 8 näkyvä funktio, send_command, antaa suoraan komennon laitteelle, joka tässä tapauksessa on show run. Show run näyttää laitteen tällä hetkellä käytössä olevan konfiguraatiot.

Toinen harjoituksissa käytetty moduuli oli ncclient, jota käytettiin NETCONFin operaatioiden yhteydessä. Sen sisäänrakennettujen funktioiden avulla käsitellä RPC-kutsuja ja XML-viestejä.

Kuvassa 10 on Ncclient:n sisäänrakennettu manager.connect()-funktio, jolla on parametrit host, port, username, password ja hostkey_verify.

```
from ncclient import manager
m=manager.connect(
    host="192.168.56.101",
    port=830,
    username="cisco",
    password="cisco123!",
    hostkey_verify=False
    )
```

Kuva 10. Ncclient:n manager.connect()-funktio.

Kuvan 10 parametrien host tarkoittaa etäyhteyden päässä olevan verkkolaitteen IP-osoitteeseen ja port on verkkoportti NETCONF-palvelulle. Username ja password ovat verkkolaitteen salasana ja käyttäjätunnus. Hostkey_verify, joka tässä tapauksessa on asetettu boolean arvolla false, on tarkoitettu varmentamaan SSH-sormenjälki /21/.

Ncclient vaatii syötettävän datan olevan XML-muotoista, jotta NETCONF ymmärtäisi mitä tehdään. Kuvassa 11 nähdään loopback-rajapinnan luonti.

```
netconf_data = """
    <config>
     <native xmlns="http://cisco.com/ns/yang/Cisco-IOS-XE-native">
      <interface>
        <Loopback>
         <name>111</name>
         <description>TEST1</description>
         <ip>
          <address>
           <primary>
            <address>100.100.100.100</address>
            <mask>255.255.255.0</mask>
           </primary>
          </address>
         </ip>
        </Loopback>
      </interface>
     </native>
    </config>
    """
```

Kuva 11. loopback-rajapinnan luonti ncclient:n avulla.

Kuten kuvasta 11 nähdään, syötettävä datan malli on XML-muotoista. Datasta löytyvät samat osuudet kuten RESTCONF-esimerkissä, mutta kuten kuvan 11 yläosasta nähdään, tehdään konfiguraatio xmlns (XML namespaces) osoittamaan kohtaan Cisco-IOS-XE-native YANG-mallin mukaisesti.

Jotta annettu data saataisiin laitteelle annettaan sille kuvan 12 mukaiset komennot. Komento edit_config on ncclient:n sisäänrakennettu komento, joka mahdollistaa tässä tapauksessa laitteen running config:n muokkauksen.

```
netconf_reply = m.edit_config(target="running", config=netconf_data)
print(xml.dom.minidom.parseString(netconf_reply.xml).toprettyxml())
```

Kuva 12. Datan välitys laitteelle

Kuvassa 12 näkyvä konfiguraatio tulostetaan käyttämällä xml.mini.dom-moduulia, joka muokkaa laitteen vastauksen XML-muotoiseksi. Laitteen vastaus nähdään kuvassa 13.

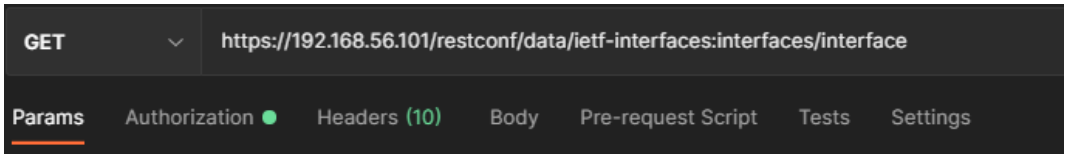
```
<?xml version="1.0" ?>
<rpc-reply xmlns="urn:ietf:params:xml:ns:netconf:base:1.0" xmlns:nc="urn:ietf:pa
rams:xml:ns:netconf:base:1.0" message-id="urn:uuid:04a47d96-4390-4bca-adad-bedb1
65af767">
        <ok/>
</rpc-reply>
```

Kuva 13. Laitteen RPC-muotoinen vastaus

Kuvassa 13 on laitteen antama vastaus konfiguraatiolle. Vastaus tulee RPC-muotoisena. Vastauksen yläosassa nähdään mitä XML-versiota on käytetty vastaukseen, tässä tapauksessa versio on 1.0. Tämän jälkeen nähdään xmlns ja message-id, nämä ovat attribuutteja, joista ensimmäinen, xmlns, määräytyy NETCONF-standardin mukaisesti. Message-id on uniikki id vastaukselle /24/. Vastauksen tärkein osuus on näiden kahden jälkeen. <ok/> viittaa siihen, että konfiguraatio on onnistunut ja laite on sen hyväksynyt.

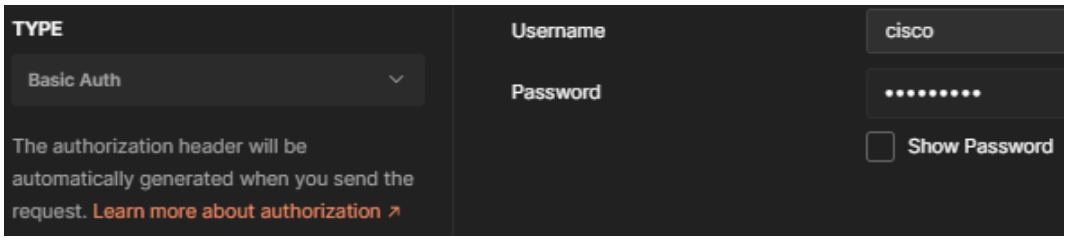
4.2 Postman-alustan käyttö verkon ohjelmoinnissa

Postman-alustalla tehdyt harjoitteet käsittelivät sitä, kuinka alustalla voitiin hakea eri rajapintojen tiedot ja kuinka muokata niitä ja kuinka luoda uusia rajapintoja. Alustaa käytettiin RESTCONF:n kanssa. Kuvasta 14 nähdään, kuinka GET-komennolla otetaan yhteyttä kuvassa näkyvään osoitteeseen.



Kuva 14. Rajapintojen kutsu Postman-alustalla

Kuvan 14 osoite sisältää laitteen IP-osoitteen, mitä protokollaa käytetään (tässä tapauksessa RESTCONF), data (tietovarasto), YANG-mallin mukainen moduuli (ieft-interfaces), YANG-mallin mukainen säiliö (interfaces) ja YANG-mallin mukainen lehti (interface). Näiden jälkeen voidaan spesifioida rajapinta, jota halutaan kutsua. Ilman spesifikaatioita vastauksena on listaus kaikista rajapinnoista. Kuvasta nähdään myös, että yhteydenottoon tarvitaan myös auktorisointi. Kuvassa 15 on Postman-alustan auktorisointivälilehti ja sen sisältö.



Kuva 15. Auktorisointi Postman-alustalla

Kuvassa 15 tyyppinä auktorisoinnille on Basic Auth, jolle voidaan antaa laitteen SSH-yhteyden salasana ja käyttäjänimi.

Kuvasta 16 nähdään laitteen vastaus GET-pyyntöön.

Kuva 16. Laitteen vastaus GET-pyyntöön

Kuvassa 16 on yksi laitteen rajapinnoista. Siitä nähdään rajapinnan nimi, rajapinnan IP-osoite ja maski, sekä onko rajapinta aktiivinen vai ei.

5 PROJEKTIN TUOTOKSET

Tuotoksena on reitittimen määritys Python-ohjelmointikielellä sekä määritysten muokkaus että uusien rajapintojen luominen Postman-ohjelmalla käyttäen REST-CONFia ja YANG-mallia.

5.1 Reitittimen asetusten määritys Python-ohjelmointikielellä

Reitittimen määrittäminen Python-ohjelmointikielellä perustuu tietämykseen Cisco CLI -komennoista. Se ei tuo paljoakaan uutta tietoa kokeneille järjestelmänvalvojille, ja vaatii laitteeseen esimääritettyä SSH-yhteyttä. Se kuitenkin nopeuttaa asetusten ajoa laitteelle ja ei vaadi erillistä PuTTY- tai muuta vastaavaa yhteyttä laitteeseen. Laitteen ei myöskään tarvitse olla fyysisesti samassa tilassa käyttäjän kanssa. Yhdellä ohjelmalla voidaan samalla kertaa muokata salasanoja, luoda uusia rajapintoja, muokata IP-osoitteita ja kaikki muu tarpeellinen. Ohjelma ajetaan ja tarkistetaan, että kaikki muokkaukset ovat menneet läpi. Tarvittaessa tarkistukseen voidaan käyttää toista ohjelmaa, joka on tarpeellista varsinkin, jos salasanoja ja IP-osoitteita on muokattu. Työkaluna tässä on käytetty Python 3.9 IDLE-kehitysalustaa ja Visual Studio Code-kehitysalustaa.

Python-ohjelman pohjana on käytetty Cisco Networking Academyn CCNA1 v7 laboratoriotyötä 10.4.4, josta käytetään reitittimen määritykseen tehtyä osuutta. Laboratoriotyössä vaaditaan laitteen nimen, salasanojen, bannerin ja IP-osoitteiden muokkausta: Poikkeuksena on kuitenkin, ettei laitteen VTY-salasanaa tai kellonaikaa muokattu. Laitteelle ei annettu IPv6-osoitetta tässä vaiheessa. Laitteen määritykseen käytetty ohjelma nähdään kuvassa 17. Kuvan yläosan koodi mahdollistaa yhteyden oton laitteeseen. Keskiosassa kuvaa on CLI-komennot, jotka muokkaavat laitteen konfiguraatioita ja lopuksi kuvan alaosassa oleva osuus mahdollistaa muokkausten läpimenon ohjelmaa ajettaessa. Huomattavaa on kuitenkin se, että ohjelma antaa virheilmoituksen, joka on kuitenkin normaalia. Laitteen IP-osoite on muuttunut, ja koodin viimeinen lausepari yrittää tulostaa laitteen uutta konfiguraatiota, mutta IP-osoitteen muutoksen tultua voimaan ohjelma ei enää saa yhteyttä laitteeseen. Tämän vuoksi suositellaan kahta eri ohjelmaa, joista toinen ajaa

konfiguraatiot laitteeseen ja toinen varmistaa, että laite on saanut konfiguraatiot oikein. Kuvassa 17 on yhdelle reitittimelle tehty ohjelmakoodi ratkaisu.

```
from netmiko import ConnectHandler

sshCli = ConnectHandler(
        device_type='cisco_ios',
        host='192.168.56.101',
        port=22,
        username='cisco',
        password='cisco123!'
        )


config_commands = [
    'hostname R1',
    'no ip domain lookup',
    'enable secret class',
    'banner motd #Admin Access only#',
    'line con 0',
    'password cisco',
    'login',
    'exit',
    'int loopback 11',
    'ip address 192.168.0.1 255.255.255.0',
    'description demoloop',
    'exit',
    'int GigabitEthernet 1',
    'ip address 192.168.56.102 255.255.255.0',
    'no shutdown',
    'exit',
    'no service password-encryption'
 ]


output = sshCli.send_config_set(config_commands)
print("config output from device:\n{}\n".format(output))


output = sshCli.send_command("show run")
print("showing running config:\n{}\n".format(output))
```

Kuva 17. Reitittimen määritys Python-ohjelmointikielellä

Kuvan 17 mukainen ratkaisu pohjautuu annettuihin esimerkkeihin, josta varsinaista useammalle laitteelle soveltuvaa ratkaisua lähdettiin tekemään.

Monelle eri laitteelle soveltuva ratkaisu hyödyntää tiedostoista luettua dataa. Konfiguraatiot ja laitteiden IP-osoitteet voidaan lukea esim. tekstitiedostoista. Näin voidaan tiedostoja muokkaamalla tehdä muutoksia laitteille. Monelle laitteelle soveltuva ratkaisu nähdään liitteessä 1, josta on esitetty kuvassa 18 tiedostojen luku.

```
# Read router Hostnames
with open('hostnames.txt') as f:
    hostnames = f.read().splitlines()

# Read list of IP addresses from file
with open('devices_list.txt') as f:
    devices_list = f.read().splitlines()

# Read router configuration
with open('router_config.txt') as f:
    config_commands = f.read().splitlines()
```

Kuva 18. Tiedostojen luku

Kuvassa 18 ohjelmakoodi hakee kolmesta eri tiedostosta tarvittavat tiedot. Ohjelma pyörii silmukassa käyttäjän antaman laitemäärän n mukaisesti. Liitteestä 1 olevan koodin alun argumenttiosuus pyytää käyttäjää antamaan cmd-ikkunassa argumentin --count, joka viittaa käyttäjää antamaan konfiguroitavien laitteiden määrän. Ohjelma tämän jälkeen, käyttää tätä annettua lukua määrittämään tarvittavan määrän IP-osoitteita ja laitenimiä, jotka luetaan tekstitiedostoista. Kuvassa 19 on IP-osoitteiden ja hostname-muuttujien määrääminen sekä konfiguraatio tiedoston muokkaaminen.

```
# Router IP list, read by n given as user argument
router_ip = devices_list[n]
router_hostname = hostnames[n]

# Replace hostname and loopback name/IP to correct value before passing it to netmiko
config_commands[0] = router_hostname
config_commands[9] = f"int loopback {n+1}"
config_commands[10] = f"ip address 10.10.10.{10+n+1} 255.255.255.0"
```

Kuva 19. IP-osoitteiden ja hostname:n määrääminen, sekä luku listoista

Tämän jälkeen ohjelma määrittää laitteelle nimen (hostname). Ohjelma lukee reitittimen konfiguraation tiedostosta ja määrää tiedostossa olevan listan paikoille kuva 19 mukaisen muuttujat. Ohjelma ottaa seuraavaksi yhteyden annetuilla laitteen yhteystiedoilla konfiguroitavaan laitteeseen ja pyytää laitetta tekemään annetut konfiguraatiot. Kuvassa 20 on yhden konfiguroidun laitteen vastaus ohjelmaan.

```
Connected to Router R1 with IP address : 192.168.56.101
configure terminal
Enter configuration commands, one per line.  End with CNTL/Z
CSR1kv(config)#hostname R1
R1(config)#no ip domain lookup
R1(config)#enable secret class
R1(config)#banner motd #Admin Access only#
R1(config)#line con 0
R1(config-line)#password cisco
R1(config-line)#login
R1(config-line)#exit
R1(config)#service password-encryption
R1(config)#int loopback 1
R1(config-if)#ip address 10.10.10.11 255.255.255.0
R1(config-if)#no shut
R1(config-if)#end
R1#
```

Kuva 20. Laitteen vastaus ohjelmakoodiin

Kuvasta 20 nähdään mihin IP-osoitteeseen ja mihin reitittimeen ollaan, sillä hetkellä ollaan yhteydessä, kuinka esimerkiksi reitittimen hostname on vaihtunut CSR1kv:sta R1:ksi. Konfiguraatio voidaan nähdä liitteestä 2. Konfiguraatio voidaan myös nähdä kuvassa 20.

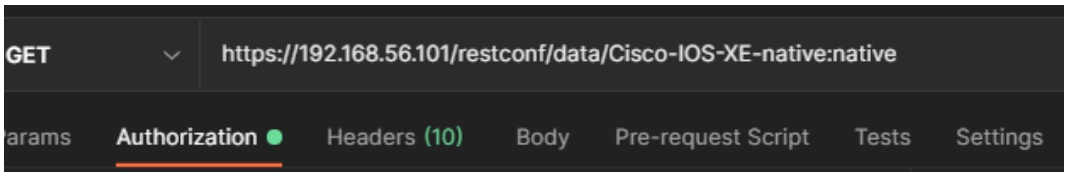
5.2 Postman-ohjelmalla tehty muokkaus ja määritys

Postman on API-kehitykseen tehty alusta, jolla voidaan muokata ja kehittää API-työkaluja /22/. Tässä sitä on käytetty yhdessä RESTCONF-protokollan ja YANG-mallin kanssa. Harjoitteissa luotiin loopback-osoitteita, jotka luodaan YANG-mallin mukaisen ieft-interfaces-moduulin alle. Tämä moduuli ei kuitenkaan tue esimerkiksi VLAN-rajapintojen tekoa, sillä nämä tarvitsevat kapselointia, jota ei voida tämän moduulin alla antaa. Moduulilla voidaan antaa IPv6-osoitteita, mutta sillä ei voida aktivoida näiden osoitteiden käyttöä. Tähän tarvitaan eri moduuli.

YANG-moduuleita on monia ja niillä on eri käyttökohteita. Näistä monista moduuleista Cisco-IOS-XE-native on moduuli, jossa nämä kumpainenkin tapaus voidaan hoitaa.

Cisco-IOS-XE-native on moduuli, joka sisältää laitteen konfiguraation. Sen kautta voidaan löytää ja vaihtaa laitteen hostname, sillä voidaan luoda uusia rajapintoja,

kuten VLAN ja mahdollistaa IPv6-osoitteiden käyttö. Kuvassa 21 on Cisco-IOS-XE-native-moduulin hakeminen GET-komennolla.



Kuva 21. Cisco-IOS-XE-native YANG-moduuli

Kuvan 21 mukainen GET-komento tuo laitteen sen hetkisen konfiguraation, jossa näkyvät kaikki laitteessa olevat rajapinnat, laitteen OS-versio, hostname, verkko-tunnus, ja yms. VLAN-rajapinnan luonti tämän moduulin kautta vaatii seuraavan kuvan mukaisen muokkauksen Postman-alustan Body-välilehdelle.

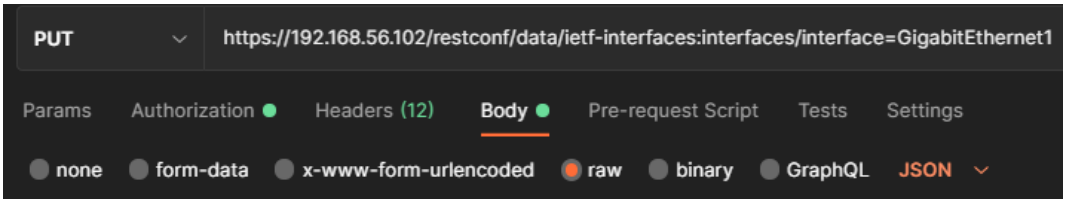
Kuvassa 22 on VLAN-rajapinnan asettelurunko.

Kuva 22. VLAN-rajapinnan luonti

Jotta kuvan 22 mukainen asettelu saadaan näkyviin, joudutaan ensin luomaan laitteelle käsin yhden VLAN-rajapinnan. Käyttämällä tätä yhtä käsin luotua rajapintaa mallina, saadaan kuvassa 22 oleva runko. Tämä lähetetään laitteelle käyttämällä

https://192.168.56.102/restconf/data/Cisco-IOS-XE-native:native/interface-osoitetta Postman-alustalla ja käyttämällä PATCH-komentoa. PATCH-komennon käyttö PUT-komennon tilalla, johtuu siitä, että nämä VLAN-rajapinnat ovat ns. alirajapintoja eikä täysin uusia rajapintoja. PATCH-komentoja käytetään päivittämään jo olemassa olevia tietoja. Samalla komennolla voidaan päivittää laite käyttämään IPv6-osoitteita.

IPv6-osoitteiden lisäys voidaan hoitaa samalla tavalla kuin normaalien rajapintojen luonti ja muokkaus. Kuvan 23 mukaista osoitetta käytetään rajapinnan GigabitEthernet1 hakemiseen.



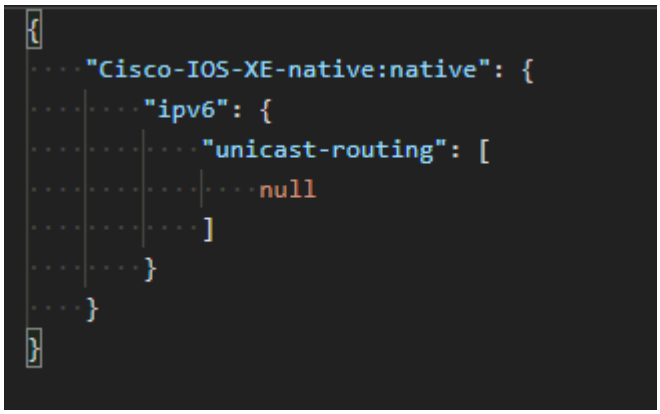
Kuva 23. GigabitEthernet1-rajapinnan muokkaus

Kuvassa 23 on GigabiteEthernet1-rajapinnan suora osoite, jolla voidaan osoitteen edellä olevaa komentoa muuttamalla, joko hakea tietoa pelkästään tästä rajapinnasta tai kuten kuvassa muokata rajapinnan tietoja. Muokkaukseen tarvitaan alla olevan kuvan 24 mukaista muokkausta Body-välilehdelle.

Kuva 24. IPv6-osoitteen lisäys GigabitEthernet1 rajapinnalle

Kuvan 24 mukaisen konfiguraation avulla voidaan antaa rajapinnalle IPv6-osoite.

Tällä tavalla voidaan antaa ja muokata olemassa olevien rajapintojen osoitteita sekä luoda uusia samankaltaisia rajapintoja. Aiemmin annettua IPv6-osoitetta ei kuitenkaan voida antaa ennemmin kuin IPv6-osoitteet on otettu käyttöön. Käyttöönotto voidaan tehdä Cisco-IOS-XE-native-moduulin alla ja siihen käytetään seuraavanlaista osoitetta: https://192.168.56.102/restconf/data/Cisco-IOS-XE-native:native. Kuvan 25 mukaista konfiguraatiota käytetään unicast-routing-komennon käyttöönottoon.



Kuva 25. IPv6-osoitteiden käyttöönotto

Kuvan 25 mukainen konfiguraatio Postman-alustan Body-välilehdellä ajetaan PATCH-komennolla laitteelle aiemmin mainittuun osoitteeseen ja tämä mahdollistaa IPv6-osoitteiden käytön laitteessa.

6 JOHTOPÄÄTÖKSET JA POHDINTA

Tämän opinnäytetyön tavoitteena oli tutustua verkon ohjelmoitavuuteen ja luoda ratkaisu, jolla voitaisiin muokata useampaa laitetta saman ajon aikana. Lisäksi opinnäytetyön aikana tutustuttiin YANG-moduuleihin ja niiden käyttöön laitekonfiguraatioissa.

Opinnäytetyön aikana saatiin aikaan ohjelmakoodi, jonka avulla voidaan muokata useampaa verkkolaitetta samanaikaisesti. Ohjelmakoodi hakee käyttäjän antaman laitemäärän mukaisen määrän IP-osoitteita ja isäntänimiä tekstitiedostoista ja lopulta tekee erillisen tekstitiedoston mukaisen konfiguraation laitteelle. Opinnäytetyön aikana tutkittiin myös YANG-moduuleja, joiden avulla tehtiin muutoksia laitekonfiguraatioihin Postman-alustan avulla. Alustan avulla muokattiin muun muassa laitteen IPv6-osoitteita ja isäntänimiä, sekä annettiin laitteelle VLAN-rajapinta.

Vaikkakin opinnäytetyön aikana aikaan saatu ohjelmakoodi oli varsin yksinkertainen, se toi kuitenkin uuden näkökulman laitteiden konfiguraatioiden muokkaukseen. Cisco CCNA-kurssien aikana kaikki laitekonfiguraatiot tehdään suoraan laitteelle käsin ja ohjelmakoodi nopeuttaa tätä osuutta hyvinkin paljon. Ohjelmakoodia voitaisiin vielä kehittää eteenpäin luomalla käyttäjäystävällisempi ratkaisu, esimerkiksi graafinen käyttöjärjestelmä. Koodia voitaisiin myös muokata tukemaan erilaisia tiedostoja kuten CSV.

Aiheena verkkojen ohjelmoitavuus on mielenkiintoinen, sillä sen avulla voidaan luoda monen näköisiä ratkaisuja laitteiden muokkaukseen. Jokainen voi luoda omanlaisensa ja omaan tarpeeseensa sopivan ratkaisun. YANG-mallien tutkiminen ja niistä oikean moduulin löytyminen oli turhauttavaa, mutta hyvin sopivan haastavaa. Moduuleja on hyvin monta ja niistä useat ovat hyvin pitkiä, joten moduulien tutkiminen vaati tarkkavaisuutta.

Järjestelmänvalvojat voivat luoda omia ratkaisujaan automatisoimaan heidän organisaationsa verkkoratkaisut tai valita käyttöönsä jo valmiita SDN-ratkaisuja. Yksinkertaisimmillaan ratkaisu voi olla ohjelmakoodin pätkä, joka ajetaan laitteelle

verkon yli suojatulla yhteydellä. Ratkaisu voi olla myös hyvin yksityiskohtainen ja monimutkainen, jolla voidaan kontrolloida ja muokata monia laitteita samanaikaisesti.

LÄHTEET

/1/ Radford, A., Maccioni, F., Zapodeanu, G., Koren, I., McLaughlin, J., Cohoe, J., Yang, J., Kotha, K., Michraf, N. & Grasby, R. IOS XE Programmability: Automating Device Lifecycle Management. Viitattu 27.04.2020. https://www.cisco.com/c/dam/en/us/products/collateral/enterprise-networks/nb-06-ios-xe-prog-ebook-cte-en.pdf

/2/ Control and Data Plane, Network Direction-verkkosivut. Viitattu 27.04.2020. https://networkdirection.net/articles/network-theory/controlanddataplane/

/3/ Tampere University of Technology, OHJ-5201-kurssimateriaali 2011. REpresentational State Transfer (REST). Viitattu 27.04.2020. http://www.cs.tut.fi/kurssit/OHJ-5201/materiaali/9.pdf

/4/ Internet Engineering Task Force (IETF). YANG - A Data Modeling Language for the Network Configuration Protocol (NETCONF). Viitattu 4.1.2020. http://www.rfc-editor.org/rfc/rfc6020.txt

/5/ Internet Engineering Task Force (IETF). Network Configuration Protocol (NETCONF). Viitattu 1.2.2020. https://tools.ietf.org/html/rfc6241#page-10

/6/ tail-f a Cisco company-verkkosivut. NETCONF overview. Viitattu 16.03.2021. https://www.tail-f.com/what-is-netconf/

/7/ Internet Engineering Task Force (IETF). RESTCONF Protocol. Viitattu 27.04.2020. https://tools.ietf.org/html/rfc8040

/8/ Donato, R. 2017. What is RESTCONF? Viitattu 16.03.2021. https://www.fir3net.com/Networking/Protocols/what-is-restconf.html

/9/ Cisco Press 2016. Network Programmability Basics. Viitattu 27.04.2020. https://www.networkcomputing.com/networking/network-programmability-basics

/10/ Cisco CCNA 3 v7.0 Enterprise Networking, Security and Automation, Viitattu 27.04.2020

/11/ Cisco CCNA 3 v7.0 Enterprise Networking, Security and Automation. Viitattu 27.04.2020

/12/ Technopedia-verkkosivut. Authentication Authorization and Accounting (AAA). Viitattu 27.04.2020. https://www.techopedia.com/definition/24130/authentication-authorization-and-accounting-aaa

/13/ Lange, K. 2016. THE LITTLE BOOK ON REST SERVICES. Viitattu 27.04.2020. https://www.kennethlange.com/books/The-Little-Book-on-REST-Services.pdf

/14/ Claise, B. 2017. YANG Opensource Tools for Data Modeling-driven Management. Viitattu 28.04.2020. https://blogs.cisco.com/getyourbuildon/yang-open-source-tools-for-data-modeling-driven-management

/15/ Donato, R. 2017. An Introduction to NETCONF/YANG. Viitattu 28.04.2020. https://www.fir3net.com/Networking/Protocols/an-introduction-to-netconf-yang.html

/16/ Donato, R. 2017. What is RESTCONF? Viitattu 29.04.2020. https://www.fir3net.com/Networking/Protoctols/what-is-restconf.html

/17/ Postman Inc. Postman API Client. Viitattu 29.04.2020. https://www.postman.com/product/api-client

/18/: Python Software Foundation. IDLE. Viitattu 29.04.2020. https://docs.python.org/3/library/idle.html

/19/ Horwitz, L. Enterprise Networks - Network programmability and automation usher in efficiency revolution. Viitattu 29.04.2020. https://www.cisco.com/c/en/us/solutions/enterprise-networks/network-programmability-automation.html

/20/ Byers, K. Netmiko-dokumentaatio. Viitattu 30.04.2020. https://ktbyers.github.io/netmiko/

/21/ Bhushan, S.; 2011–2014. Ncclient-dokumentaatio. Viitattu 30.04.2020. https://ncclient.readthedocs.io/en/latest/index.html

/22/ Postman Inc. The Postman API Platform. Viitattu 08.02.2021. API Platform: API Tools & Solutions | Postman

/23/ Ncclient-dokumentaatio. Viitattu 09.02.2021. GitHub - ncclient/ncclient: Python library for NETCONF clients

/24/ Clark, S. 2020. Introduction to Programmability – Part 2. viitattu 16.02.2021. Introduction to Programmability – Part 2 (cisco.com)

/25/ Cisco Networking Academy. Model Driven Programmability, Chapter 2: YANG, RESTCONF & NETCONF. s.13 Viitattu 16.03.2021

/26/ Sudo Null Company 2012. Netconf. Start. Viitattu 16.03.2021. https://sudonull.com/post/149378-NETCONF-Start

/27/ Ciscolive!-verkkosivut. Introduction to YANG. Viitattu 16.03.2021- http://yang.ciscolive.com/pod0/labs/lab2/lab2-m1

/28/ JD’s Notepad -verkkosivut. RESTCONF. Viitattu 17.03.2021. https://jdsnotepad.wordpress.com/2018/10/29/restconf/

LIITE 1 Ohjelmakoodi

```
import argparse
from netmiko import ConnectHandler



# Command line argument for amount of routers user can choose
def get_cmd_line_args():
    parser = argparse.ArgumentParser(description="Router configura-
tion settings")
    parser.add_argument("--count",
                        default="1",
                        type=int,
                        help="Give an amount of routers to send con-
figuration to as CLI parameter",
                        dest="count")
    return parser.parse_args()


# Function which have tester, recieves n as user argument
def tester(n):
    # Read router Hostnames
    with open('hostnames.txt') as f:
        hostnames = f.read().splitlines()

    # Read list of IP addresses from file
    with open('devices_list.txt') as f:
        devices_list = f.read().splitlines()

    # Read router configuration
    with open('router_config.txt') as f:
        config_commands = f.read().splitlines()

    # Router IP list, read by n given as user argument
    router_ip = devices_list[n]
    router_hostname = hostnames[n]

    # Replace hostname and loopback name/IP to correct value be-
fore passing it to netmiko
    config_commands[0] = router_hostname
    config_commands[9] = f"int loopback {n+1}"
    config_commands[10] = f"ip ad-
dress 10.10.10.{10+n+1} 255.255.255.0"

    # Device connec-
tion info passed to netmiko for log in into router
```

```
    device_info = {
        'device_type':'cisco_ios',
        'host':router_ip,
        'port':22,
        'username':'cisco',
        'password':'cisco123!'
        }

    # Netmiko library connection and sending config to router
    device_connect = ConnectHandler(**device_info)
    output = device_connect.send_config_set(config_commands)

    print(f"Connected to Router",router_hostname[-2:] ,"with IP ad-
dress :", router_ip)
    print(output)



def main(args):
    # Initiate empty task list
    tasks = []
    # Run n amount of tasks based on user argument list
    for n in range(0, args.count):
        tasks.append(tester(n))



if __name__ == "__main__":
    args = get_cmd_line_args()
    main(args)
```

LIITE 2 Konfiguraatiotiedosto

hostname

no ip domain lookup

enable secret class

banner motd #Admin Access only#

line con 0

password cisco

login

exit

service password-encryption

loopback

ip

no shut