Automaattinen päätelaitteiden konfigurointi ohjelmisto

HAMK
HÄMEEN AMMATTIKORKEAKOULU
HÄME UNIVERSITY OF APPLIED SCIENCES

Ammattikorkeakoulututkinnon opinnäytetyö

Riihimäki, tieto- ja viestintätekniikka

syksy, 2018

Joonas Salo

HÄMEEN AMMATTIKORKEAKOULU
HÄME UNIVERSITY OF APPLIED SCIENCES

Tieto- ja viestintätekniikka
Riihimäki

Tekijä Joonas Salo **Vuosi** 2018

Työn nimi Automaattinen päätelaitteiden konfigurointi ohjelmisto

Työn ohjaaja/t Petri Kuittinen HAMK, Toni Lappalainen Elisa Oyj

TIIVISTELMÄ

Tämän opinnäytetyön tilaaja on Elisa Oyj.

Tämän opinnäytetyön aiheena oli automatisoida Elisa Oyj:n yritysasiakkaiden päätelaitteiden (CPE) käyttöönotto. Työn tavoitteeksi muodostui palvelimella automaattisesti ajettava tietokoneohjelmisto, joka ohjelmoitaisiin Python -ohjelmointikielellä.

Opinnäytetyön tarkoituksena oli vähentää yritysasiakkaiden päätelaitteiden käyttöönottoon kuluvaa aikaa ja helpottaa sekä selkiyttää kenttäasentajan työskentelyä asiakkaan toimipisteessä. Ohjelmistosta oli tarkoitus tehdä myös skaalautuva tulevaisuutta varten, jotta siihen voisi helposti tehdä uusia toiminnallisuuksia ja testejä uusia päätelaitteita, asiakkaita ja verkkoja varten.

Opinnäytetyö aloitettiin suunnittelupalaverilla Elisa Oyj:n kehityspäällikön kanssa, jotta saatiin selkeä kuva mitä asioita ohjelmiston pitäisi tehdä. Päädyttiin ratkaisuun, jossa on 4 testausoperaatiota sekä varsinainen päätelaitteen konfiguraatio-operaatio. Työtä varten suunniteltiin ja toteutettiin myös tietokanta konfiguroitaville päätelaitteille käyttäen MariaDB -tietokantaohjelmistoa.

Opinnäytetyön tuloksena syntyi automaattisesti toimiva tietokoneohjelmisto, joka suunniteltujen testien jälkeen konfiguroi asiakkaan päätelaitteen käyttövalmiiksi viimeistelyprosessissa.

Avainsanat Automatisointi, Ansible, Git, ohjelmointi, tietoverkko, tietoliikenne, Python, tietoliikennelaite, tietokanta, MySQL, MariaDB, Linux

Sivut 28 sivua

HÄMEEN AMMATTIKORKEAKOULU
HÄME UNIVERSITY OF APPLIED SCIENCES

Information- and Communication Technology
Riihimäki

Author Joonas Salo **Year** 2018

Subject Automated CPE configuration software

Supervisors Petri Kuittinen HAMK, Toni Lappalainen Elisa Plc

ABSTRACT

This project was commissioned by Elisa Plc.

The subject of this thesis was to automate customer premises equipment (CPE) deployment of the corporate customers of Elisa Plc. A computer program running automatically on a server and programmed in the Python -programming language formed the goal of this project.

The purpose of this project was to reduce the time that was spent in the deployment of the customer premises equipment of corporate customers. An additional purpose was to clarify field engineer's work and make the work easier on customer premises.

The program itself was meant to be written scalable for the future, so that different tests and functionalities could be added to it for possible new CPE, customers and networks.

Working with the thesis started at a planning meeting with the development manager of Elisa Plc to get a clear idea of which functions the program should possess. The program was to perform three test operations before the main CPE configuration operation and one final reachability test operation. The database was also designed and created for the program using MariaDB -relational database management system.

The outcome of the project was software running on a server that after tests configured CPE of the corporate customers automatically to the deployment process.

Keywords Automatization, Ansible, Git, programming, network, communication, Python, CPE device, database, MySQL, MariaDB, Linux

Pages 28 pages

SISÄLLYS

Lyhenteet

API	Application Programming Interface. Ohjelmointirajapinta.
CPE	Customer-premises equipment. Asiakkaan päätelaite
DSL	Digital subscriber line. Digitaalinen tilaajayhteys. Tietoliikennetekniikka, jossa tietoliikenne kulkee lankapuhelinverkossa.
GSM	Global System for Mobile communications. 2. sukupolven matkapuhelinverkkojärjestelmä.
ICT	Information and Communication Technology. Tieto- ja viestintäteknologia ala
OID	Object Identifier. Yksilöintitunnus. Yleiskäyttöinen kansainvälisesti vain yhteen kohteeseen liitettävä numerosarja, joka yksilöi kyseisen kohteen yksiselitteisesti ISO/IEC 8824-1:2002 -standardin mukaisesti. Käytetään SNMP -operaatioissa kohdistamaan laitteelta haettu tai muutettu tieto.
PE	Provider Edge. Operaattorin tietoliikenneverkon reunareititin.
SDN	Software-defined networking. Ohjelmisto-ohjatut verkot. Teknologia, missä ohjelmistoilla ohjataan minkä tahansa toimittajan tietoverkko elementtejä.
SNMP	Simple Network Management Protocol. Tietoverkkolaitteiden hallinnointiin käytettävä tietoliikenneprotokolla.
SSH	Secure Shell. Salattuun tietoliikenteeseen tarkoitettu protokolla.
SQL	Structured Query Language. IBM:n kehittämä standardoitu tietokantojen kyselykieli. Käytännössä oletus kyselykieli kaikissa tietokantaohjelmistoissa. SQL-clause tarkoittaa SQL -kyselykielellä tehtyä operaatiota.
TCL	Tool Command Language. Tulkattava ohjelmointikieli. Yleisesti käytetty tietoverkkolaitteiden automatisointiin etenkin CISCO:n laitteiden.
TFTP	Trivial File Transfer Protocol. Tiedostojen siirtoon tarkoitettu protokolla.
WAN	Wide Area Network. Laajaverkko. Tiedonsiirtoverkko, joka peittää laajoja maantieteellisiä alueita. Yhdistää lähiverkot sekä kaupunkiverkot suuremmaksi verkoksi. Yleensä tarkoitetaan operaattorin tietoliikenneverkkoa.
YAML	YAML Ain't Markup Language tai alunperin Yet Another Markup Language. Konfiguraatiotiedostoissa usein käytetty merkintäkieli. Ansible -konfiguraatiotyökalun oletus merkintäkieli.

1 JOHDANTO

Elisa Oyj:ssä oli tarve automatisoida yritysasiakkaiden liittymien käyttöönotto eli päätelaitteiden konfigurointi. Opinnäytetyön tavoitteeksi muodostui Python -ohjelmointikielellä kirjoitettava tietokoneohjelmisto, joka toimisi palvelimella ja konfiguroisi päätelaitteet automaattisesti.

Esittelen tässä dokumentaatiossa opinnäytetyön lähtökohdat ja toteutuksen sekä lopputuloksen ja kehitysmahdollisuudet. Lähtökohdista esittelen ensimmäisenä työn tilaajan eli Elisa Oyj:n taustaa sekä tilaajan kanssa sovitut tavoitteet. Kerron myös mitä järjestelmiä ja työkaluja opinnäytetyön tekemiseen käytettiin. Viittaan tässä dokumentaatiossa myös usein käsitteisiin päätelaite ja käyttöönotto.

Päätelaitteella tarkoitetaan tässä dokumentaatiossa yritysasiakkaan tiloissa olevaa reititintä (CPE), joka mahdollistaa yritysasiakkaan pääsyn Elisa Oyj:n runkoverkkoon. Yritysasiakkaan reititin on yhteydessä Elisa Oyj:n runkoverkon reunareitittimeen (PE), joko suoraan tai esim. DSL-keskittimen kautta.

Käyttöönotolla tarkoitetaan tässä dokumentaatiossa yritysasiakkaan liittymän viimeistelyä niin, että tietoliikenneyhteys on valmiina käyttöön. Yritysasiakkaan liittymän käyttöönotossa kenttäasentaja asentaa päätelaitteen paikalleen, kytkee tarvittavat verkkojohdot kiinni ja tekee aloituskonfiguraation päätelaitteelle.

Esittelen tässä dokumentaatiossa myös tietoliikenneverkkojen ja päätelaitteiden automatisointia yleisesti sekä automatisointi menetelmiä ja mitä hyötyä automatisoinnista on. Toteutuksesta kerron miten ohjelmistoa ja tietokantaa suunniteltiin sekä miten niitä rakennettiin. Kerron myös ohjelmiston ja tietokannan testauksesta ja lopputuloksesta. Dokumentaation lopussa kerron opinnäytetyön yhteenvedon ja kehitysmahdollisuuksista.

Tavoitteena tässä opinnäytetyössä oli vähentää kenttäasentajien tarvetta soittaa hallintakeskukseen sekä helpottaa heidän työskentelyään automatisoimalla tämän varsinaisen päätelaitekonfiguraation lataus sekä konfiguraation käyttöönotto. Samalla toteutuva toinen tavoite oli helpottaa hallintakeskuksen tietoliikenneasiantuntijoiden ruuhka työmäärää sellaisten tietoverkkoliittymien käyttöönotossa, missä ei välttämättä tarvittu tietoliikenneasiantuntijan työpanosta ja ammattitaitoa. Samalla oli myös tarkoitus vähentää hallintakeskukseen tulevien puhelujen määrää kenttäasentajilta.

2 OPINNÄYTETYÖN LÄHTÖKOHDAT

2.1 Opinnäytetyön tilaaja

Tämän opinnäytetyön toimeksiantajana toimi Elisa Oyj, joka on suomalainen tietoliikenne-, ICT- ja digitaalisten palveluiden yritys, jonka päämarkkina-alueet ovat Suomi ja Viro. Elisa Oyj on Suomen johtava mobiili- ja kiinteän verkon operaattori, joka tarjoaa ympäristöystävällisiä palveluita viestimiseen ja viihtymiseen sekä työvälineitä organisaatioiden toiminnan digitalisoimiseen ja tuottavuuden parantamiseen. Elisa Oyj palvelee yli 2,8 miljoonaa asiakasta maailmanlaajuisesti, joilla on yli 6,2 miljoonaa liittymää. (Elisa Oyj, 2018)

Elisa Oyj on myös edelläkävijä uusien verkkoteknologioiden ja -innovaatioiden, kuten 5G-mobiiliverkkotekniikan kehittämisessä. Elisa Oyj tekee myös yhteistyötä toisten kansainvälisten tietoliikenneoperaattoreiden kanssa, kuten Vodafonen ja Telenorin, mikä mahdollistaa kansainvälisesti kilpailukykyiset palvelut sekä tekniikat. (Elisa Oyj, 2018)

Yrityksen teknisiin edistysaskeliin kuuluu maailman ensimmäinen kaupallisessa mobiiliverkossa soitettu GSM-puhelu vuonna 1991 sekä maailman ensimmäinen 3G-mobiiliverkossa soitettu puhelu vuonna 2006. Kesäkuussa 2018 Elisa Oyj ilmoitti avanneensa maailman ensimmäisen kaupallisen 5G-mobiiliverkon Tampereelle ja Tallinnaan. (Elisa Oyj, 2018)

Elisa Oyj:n liikevaihto vuonna 2017 oli 1,79 miljardia euroa ja henkilöstöä oli 4700, 13 eri maassa. Elisa Oyj on julkisesti noteerattu pörssiyhtiö Nasdaq Helsinki Suuret Yhtiöt -listalla ja osakkeenomistajia sillä on noin 190 000. (Elisa Oyj, 2018)

Elisa Oyj kehittää, optimoi ja automatisoi myös yrityksen sisäisiä järjes-telmiä ja tietoverkkoja jatkuvasti. Tämä sisäinen kehitystyö antoi myös aiheen tälle opinnäytetyölle.

2.2 **Opinnäytetyön tavoitteet**

Opinnäytetyön tavoitteena oli automatisoida Elisa Oyj:n yritysasiakkaiden tietoverkkoliittymien päätelaitteiden käyttöönotto. Päätelaitteella (CPE) tarkoitetaan tässä Opinnäytetyössä yritysasiakkaan tiloissa olevaan laitetta, joka on yhteydessä Elisa Oyj:n runkoverkon reunalaitteeseen (PE) tai jos kyseessä on mobiiliverkossa toimiva päätelaite niin silloin laite ottaa yhteyden Elisan Oyj:n mobiilitukiasemaan.

Yritysasiakkaiden tietoverkkoliittymien käyttöönotto tapahtuu pääsääntöisesti aina kenttäasentajan kanssa tehtävässä viimeistelyprosessissa. Kenttäasentaja vie uuden päätelaitteen asiakkaan tiloihin, tekee aloituskonfiguraation päätelaitteelle ja tarvittaessa soittaa Elisa Oyj:n hallintakeskukseen tietoliikenneasiantuntijalle, jos käyttöönotossa ilmenee ongelmia tai tietoverkkoliittymä on standardista poikkeava.

Tietoverkkoliittymän viimeistelyprosessissa hallintakeskuksen tietoliikenneasiantuntija tekee päätelaitteelle prosessin mukaiset testit sekä reitittää yritysasiakkaan omat tietoverkot Elisa Oyj:n runkoverkkoon, jos näin ei ole jo tehty. Tämä prosessi voidaan tehdä yksinään tietoliikenneasiantuntijan toimesta ilman, että kenttäasentaja soittaa hallintakeskukseen.

Kenttäasentajan suorittama työ käyttöönottovaiheessa alkaa päätelaitteen asennuksella asiakkaan toimipisteeseen. Tämän jälkeen päätelaitteeseen syötetään virta ja päätelaite kytketään päälle. Kun päätelaite on täydellisesti käynnistynyt niin kenttäasentaja syöttää päätelaitteelle ns. aloituskonfiguraation. Aloituskonfiguraatiossa kenttäasentaja syöttää laitteelle sen hallinta IP-osoitteen (WAN-CE) sekä staattisen reitityksen Elisan runkoverkon toiseen osoitteeseen (WAN-PE), joka on Elisa Oyj:n runkolaitteella määritettynä. Kun päätelaitteelle on syötetty tarvittava aloituskonfiguraatio niin tämän jälkeen kenttäasentaja lataa varsinaisen päätelaitekonfiguraation Elisa Oyj:n palvelimelta ja käynnistää uudelleen päätelaitteen, jonka jälkeen varsinainen konfiguraatio tulee käyttöön.

Tavoitteena tässä opinnäytetyössä oli vähentää kenttäasentajien tarvetta soittaa hallintakeskukseen sekä helpottaa heidän työskentelyään automatisoimalla tämän varsinaisen päätelaitekonfiguraation lataus sekä konfiguraation käyttöönotto. Samalla toteutuva toinen tavoite oli helpottaa hallintakeskuksen tietoliikenneasiantuntijoiden ruuhka työmäärää sellaisten tietoverkkoliittymien käyttöönotossa, missä ei välttämättä tarvittu tietoliikenneasiantuntijan työpanosta ja ammattitaitoa. Samalla oli myös tarkoitus vähentää hallintakeskukseen tulevien puhelujen määrää kenttäasentajilta.

2.3 **Opinnäytetyössä käytetyt järjestelmät ja työkalut**

Opinnäytetyön tekemiseen käytettiin monia eri järjestelmiä ja työkaluja. Opinnäytetyö aloitettiin aluksi tekijän omalla kotitietokoneella, josta se myöhemmin siirrettiin Elisa Oyj:n testipalvelimelle ja tästä se lopulta siirrettiin valmiina tuotantopalvelimelle varsinaista tuotantokäyttöä varten. Tekijän omalla kotitietokoneella testiympäristönä toimi Oraclen Virtualbox, joka on ilmainen ja avoimeen lähdekoodiin perustuva virtuaalilaiteympäristö. Oraclen Virtualbox:ssa testikäyttöjärjestelmänä toimi Linux Debian 9.0 Stretch -käyttöjärjestelmä.

Opinnäytetyön tekijän omassa testiympäristössä käytettiin myös GNS3 (Graphical Network Simulator 3) -ohjelmistoa, joka on tietoliikennelaitteiden emulointiin käytettävä ohjelmisto. Käytössä oli myös itse varsinaista opinnäytetyön ohjelmointia varten PyCharm -ohjelmisto, joka on Python -ohjelmointikieltä varten kehitetty IDE (integrated development environment) eli integroitu kehitysympäristö.

Opinnäytetyöhön täytyi myös suunnitella päätelaitetietokanta, joten tätä varten testiympäristön Linux Debian -käyttöjärjestelmässä käytettiin MariaDB -tietokantaohjelmistoa. MariaDB on MySQL -tietokantaohjelmistoon pohjautuva relaatiotietokantajärjestelmä, jota kehittää sama yhteisö kuin alkuperäistä MySQL -tietokantaohjelmistoa.

Elisa Oyj:n testiympäristönä käytettiin Linux Red Hat Enterprise -palvelinkäyttöjärjestelmää ja myös MariaDB -tietokantaohjelmistoa. Kumpaankin testiympäristöön asennettiin myös Python-tulkki, itse opinnäytetyön ohjelmiston ohjelmointia varten.

2.3.1 Oracle Virtualbox

Oracle Virtualbox käyttöjärjestelmien ja muiden ohjelmistojen virtualisoimiseen tarkoitettu tietokoneohjelmisto. Se on ilmainen ja perustuu avoimeen lähdekoodiin. Oracle Virtualbox:in voi asentaa lähes kaikkiin PC-pohjaisiin käyttöjärjestelmiin ja siihen voi asentaa lähes kaikkia käytössä olevia käyttöjärjestelmiä, kuten Linux, Windows, Solaris, BSD ja OS/2 sekä rajoitetusti Applen MacOS -käyttöjärjestelmiä.

Oracle Virtualbox valikoitui testiympäristön virtuaalialustaksi sen helppouden ja yksinkertaisuuden takia. Oracle Virtualbox oli yksinkertainen asentaa, siihen löytyi helposti ilmaisia Linux -käyttöjärjestelmän image -tiedostoja sekä se oli helppo ottaa käyttöön. Virtuaalialustalta ei myöskään vaadittu muita virtualisointiominaisuuksia tässä opinnäytetyössä kuin pelkkä Linunx -käyttöjärjestelmän käyttö.

2.3.2 Linux Debian 9.0 Stretch

Linux Debian -käyttöjärjestelmä on maailman suosituin Linux -jakelupaketti sekä ilmainen ja perustuu myös ilmaiseen lähdekoodiin. Linux Debian -käyttöjärjestelmää käytetään työasemissa sekä palvelin -tietokoneissa sekä sitä on käytetty pohjana moni muille Linux -jakelupaketeille.

Linux Debian 9.0 Stretch -käyttöjärjestelmä valikoitui opinnäytetyön virtuaalikäyttöjärjestelmäksi, koska Elisa Oyj:n testi- ja tuotantopalvelimet, joissa itse opinnäytetyön ohjelmistoa tultaisiin käyttämään, käyttivät palvelinkäyttöjärjestelminään Red Hat Enterprise Linux -käyttöjärjestelmää.

Red Hat Enterprise Linux -käyttöjärjestelmä (RHEL) on maksullinen ohjelmisto, joten tekijän omassa testiympäristössä jouduttiin käyttämään ilmaista Linux Debian -käyttöjärjestelmää. Virtuaalikäyttöjärjestelmältä ei vaadittu muita ominaisuuksia kuin Python-tulkin sekä MariaDB -tietokantaohjelmiston asentaminen ja käyttäminen.

2.3.3 Red Hat Enterprise Linux

Red Hat Enterprise Linux -käyttöjärjestelmä (RHEL) on Red Hat -yhtiön kehittämä käyttöjärjestelmä, jota käytetään pääasiassa palvelintietokoneissa. RHEL on kaupallinen ja maksullinen käyttöjärjestelmä mutta se perustuu silti avoimeen lähdekoodiin. Elisa Oyj:n testi- ja tuotantopalvelimilla on käytössä RHEL Server, joille opinnäytetyön ohjelmisto siirrettiin työn keskivaiheilla. Palvelinalustan ja käyttöjärjestelmän vaihto ei vaatinut muutoksia ohjelmiston koodiin eikä muitakaan isompia toimenpiteitä.

RHEL on Enterprise-ympäristöihin tarkoitettu palvelin -käyttöjärjestelmä, joka tunnetaan turvallisuudestaan ja luotettavuudestaan. RHEL:in innovatiivinen tukitilaus eli Subscription-malli mahdollistaa teknologiat, tuotteet ja tukipalvelut asiakasta hyödyttävällä tavalla. RHEL Server takaa erinomaisen suorituskyvyn, turvallisuuden, skaalautuvuuden ja käytettävyyden. (Red Hat Inc, 2018)

RHEL Server -ratkaisut myydään sähköisesti toimitettavina tukitilauksina, joihin asiakas voi valita seuraavan ylläpito-/tukipaketin: Self-Support (ei suositella tuotantoympäristöön) sekä Standard tai Premium joko yhdeksi tai kolmeksi vuodeksi. (Red Hat Inc, 2018)

RHEL Server -tuotteisiin on valittavissa myös lisäkomponentteja esimerkiksi järjestelmänhallintaan ja tehokkaampien tiedostojärjestelmien käyttämiseksi. Lisäksi saatavilla on tuotteet SAP-, HPC-, suurkone- ja työasemakäyttökohteisiin. (Red Hat Inc, 2018)

2.3.4 GNS3

GNS3 (Graphical Network Simulator 3) on tietoverkko- ja tietoliikennelaitteiden emulointiin tarkoitettu virtuaaliohjelmisto. GNS3 -ohjelmistossa voi emuloida, konfiguroida, testata ja suorittaa vianetsintää virtuaalisissa sekä oikeissa tietoverkoissa.

Alussa GNS3 tuki vain Cisco:n kytkimien ja reitittimien emulointia Dynamips-lisäosan kautta mutta nyt sillä voi emuloida muitakin laitteita kuten: Cisco:n virtuaalisia kytkimiä, Cisco:n ASA -palomuureja, Brocade:n V -reitittimiä, Docker -instansseja, Linux -käyttöjärjestelmiä sekä HPE -tietoliikennelaitteita. GNS3 -ohjelmistossa käytetään oikeita Cisco:n IOS -käyttöjärjestelmän imagetiedostoja.

GNS3 -ohjelmiston voi asentaa suoraan työasemalle kokonaisuudessaan, kuten tässä opinnäytetyössä tehtiin, jolloin itse ohjelmisto ja palvelin toimivat samalla työasemalla. Sen voi myös asentaa paikalliselle virtuaalipalvelimelle tai etänä olevalle palvelimelle. GNS3 -ohjelmiston käyttö jäi tässä opinnäytetyössä vähäiseksi, johtuen työn melko nopeasta siirrosta toimeksiantajan testipalvelimelle.

2.3.5 MariaDB

MariaDB on ilmainen ja avoimeen lähdekoodiin perustuva relaatio tietokanta järjestelmä, jota kehittää sama yhteisö kuin alkuperäistä MySQL -tietokantaohjelmistoa. Sun Microsystems osti MySQL AB:n vuonna 2008 ja vuonna 2009 Oracle Corporation osti Sun Microsystems:in, jolloin MySQL:n oikeudet siirtyivät Oracle Corporationille ja ohjelmisto muuttui maksulliseksi. MySQL:n alkuperäinen kehittäjä Michael ”Monty” Widenius alkoi vuonna 2010 kehittämään ilmaista ja avoimeen lähdekoodiin perustuvaa korvaajaa MySQL -tietokantaohjelmistolle ja kehitti MariaDB:n.

MariaDB valikoitui opinnäytetyön päätelaitteiden tietokanta järjestelmäksi, koska siinä on täysin sama SQL -syntaksi MySQL:n kanssa sekä samat ohjelmointirajapinnat (eng. API – application programming interface) ja kirjastot. MariaDB on myös nykyään monien Linux -jakelupakettien kanssa oletuksena asennettava tietokanta järjestelmä, sekä opinnäytetyön tekijällä on MariaDB/MySQL -tietokannoista eniten kokemusta.

2.3.6 PyCharm

PyCharm on erityisesti Python -ohjelmointikielelle suunniteltu ja kehitetty integroitu kehitysympäristö (IDE). Sen on kehittänyt tšekkiläinen yhtiö nimeltä JetBrains. PyCharm:n ominaisuuksiin kuuluu ohjelmistokoodin analysointi, graafinen vianetsintä, integroitu testaus, integroitu versionhallinta ja tuki web-ohjelmointikehys Django:lle.

PyCharm:sta on saatavilla ilmainen Community Edition versio ja maksullinen Professional versio, joista molemmat ovat asennettavissa Windows, MacOS ja Linux -käyttöjärjestelmille.

PyCharm Professional -ohjelmistoa käytettiin aluksi tekijän omassa testiympäristössä itse opinnäytetyön ohjelmointiin sen kattavien ominaisuuksien takia mutta tämänkin käyttö jäi vähäiseksi, kun ohjelmakoodi siirrettiin virtuaalialustalle melko aikaisessa vaiheessa.

2.3.7 Python

Python on monipuolinen ja tulkattava korkean tason ohjelmointikieli. Python -ohjelmointikielen loi Guido Van Rossum vuonna 1991, jonka tarkoituksena oli luoda helposti ohjelmoitava ja luettava ohjelmointikieli. Python -ohjelmointikieltä pidetäänkin helppona oppia sen yksinkertaisen syntaksin ja korkean tason tietorakenteiden takia sekä sitä suositellaan usein ensimmäiseksi ohjelmointikieleksi aloittelijalle.

Tulkattavalla Python -ohjelmointikielellä kirjoitetut ohjelmat ovat valmiita ajettaviksi välittömästi, eikä niitä tarvitse ensin kääntää. Tällä tavalla ohjelmoimisen, vianetsinnän ja testaamisen voi tehdä lyhyissä sykleissä ja nopeasti.

Python valikoitui tämän opinnäytetyön ohjelmointikieleksi sen kattavien tietoverkko-ohjelmointiin tarkoitettujen moduulikirjastojen sekä MariaDB -rajapinnan takia. Elisa Oyj:n testi- ja tuotantopalvelimilla käytössä oleva Ansible -automatisointityökalu on myös toteutettu Python -ohjelmointikielellä, joten ohjelmiston täytyi olla yhteensopiva tämän kanssa. Python -tulkista käytettiin versiota 2.7 molemmissa testiympäristöissä.

2.4 **Opinnäytetyön suunnittelu**

Opinnäytetyön suunnittelu aloitettiin suunnittelupalaverilla Elisa Oyj:n kehityspäällikön kanssa ja palavereita jatkettiin koko projektin ajan, alussa kerran viikossa ja projektin loppuvaiheessa tarvittaessa.

Suunnittelussa kartoitettiin ensin mitä ohjelmiston pitäisi tehdä ennen varsinaista päätelaitteen konfigurointia sekä miten itse konfigurointi operaatio suoritettaisiin. Suunnittelussa tultiin johtopäätökseen, että päätelaitteille olisi tehtävä erilaisia testejä ennen kuin niitä voisi luotettavasti konfiguroida. Kaikkiaan erilaisia testausoperaatioita suunniteltiin kolme ennen varsinaista konfigurointioperaatiota sekä yksi testausoperaatio konfiguroinnin jälkeen, jotta voitiin varmistua, että konfigurointi olisi suoritettu onnistuneesti.

Palaverissa todettiin myös, että päätelaitteille suunniteltaisiin tietokanta, jotta niille tehtäviä testejä voitaisiin tarkastella sekä käyttöönotettavat päätelaitteet lisätä tähän tietokantaan.

3 TIETOLIIKENNEVERKON AUTOMATISOINTI

3.1 **Yleistä**

Tietoliikenneverkon automatisointi tarkoittaa tilannetta, missä ohjelmisto ja/tai skripti automaattisesti konfiguroi, provisioi, hallinnoi ja/tai testaa tietoverkkolaitetta ilman asiantuntijan manuaalisia toimenpiteitä. Tietoliikenneverkon automatisointi on suuressa kasvussa tällä hetkellä, erityisesti suurissa yrityksissä ja tietoliikenneoperaattoreilla.

Tietoliikenneverkot ovat nykyään erittäin kriittinen osa yritysten liiketoimintaa, joissa pienimmätkin häiriöt ja viat voivat aiheuttaa liiketoiminnalle suuriakin menetyksiä. Tietoliikenneoperaattoreilla nämä häiriöt ja viat korostuvat vielä enemmän niiden koskettaessa mahdollisesti, satoja ellei tuhansia asiakkaita.

Tietoliikenneverkkojen konfigurointi, ylläpito ja vikatilanteista toipuminen suoritetaan vielä nykyäänkin useissa yrityksissä manuaalisesti. Yksinkertaisimmat toiminnotkin, kuten tietoverkkolaitteiden asetusten varmuuskopioinnit saatetaan suorittaa manuaalisesti. Joskus nämä manuaaliset toimenpiteet saatetaan joutua suorittamaan paikan päällä, jotta yrityksen tietoliikenneverkko voidaan pitää toimintakykyisenä, eikä siihen tule liiketoimintaa häiritseviä katkoksia.

Nykyään nämä manuaalisesti tehtävät toimenpiteet voidaan suorittaa tietoliikenneverkossa automaattisesti ilman, että asiantuntijan tarvitsee puuttua niihin. Hyvin suunnitellussa ja toteutetussa automatisoidussa ympäristössä voidaan esim. palautua runkolaiteviasta ilman, että tietoliikenneverkkoon tulee katkosta ollenkaan.

3.2 **SDN**

SDN (eng. software-defined networking) on arkkitehtuuri, jonka tarkoituksena on tehdä tietoliikenneverkkojen käyttöönotosta, konfiguroinnista, ylläpidosta ja skaalautuvuudesta entistä tehokkaampaa ja joustavampaa.

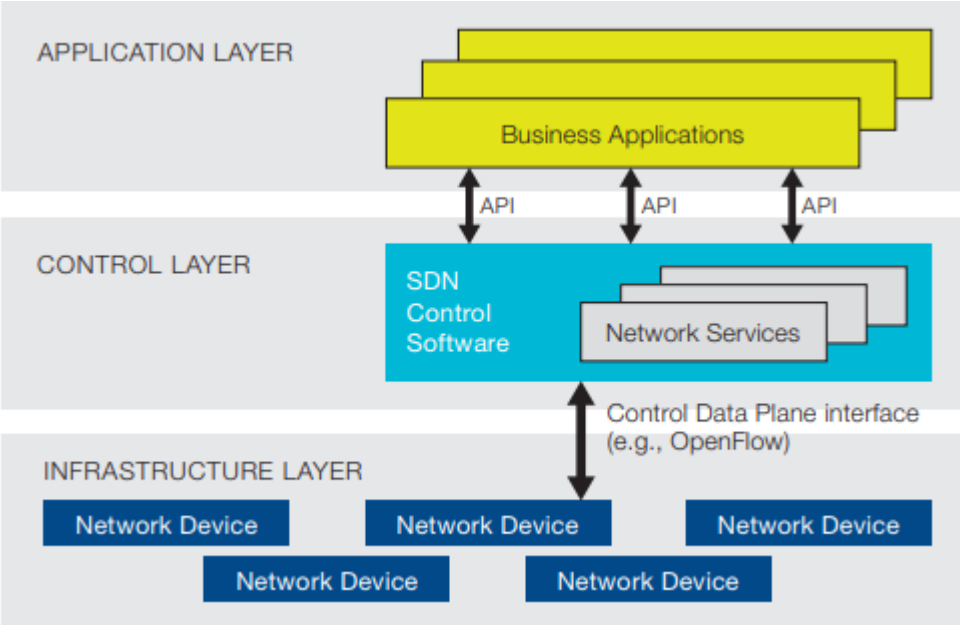
SDN arkkitehtuurissa on eroteltu hallinta- ja datatasot toisistaan. Tämän etuna tietoliikenneverkon hallinta ja ylläpito voidaan keskittää yhteen loogiseen paikkaan, jonka ansiosta tietoliikenneverkon automatisointi ja skaalautuvuus on tehokkaampaa.

Perinteisessä tietoliikenneverkon arkkitehtuurissa verkkorakenne toteutetaan hierarkkisesti. Verkkorakenne muodostuu puumallisesta verkkotopologiasta, jossa ethernet-kytkimiä on kerroksittain. Tämä arkkitehtuuri sopi, kun tiedonsiirto käytiin työasema/palvelin -tyyppisessä tiedonsiirrossa. (ONF, 2012, s. 3)

Nykypäivän yrityksen tiedonsiirrossa työasemat muodostavat useita yhteyksiä tietokantoihin ja palvelimiin. Tämä luo haasteita nykypäivän tietoliikenneverkon arkkitehtuurille, kun tietoa voidaan kerätä ja siirtää usean palvelimen välillä luoden useita tietoliikenneyhteyksiä, ennen kuin käyttäjä pääsee tietoon käsiksi. Tämän vuoksi yritykset harkitsevat tiedon käsittelyyn ja sen säilytykseen pilvipohjaista ratkaisua, jolla voidaan vähentää yhden käyttäjän luomaa tietoliikennemäärää. (ONF, 2012, s. 3)

Tietoliikenneverkkojen ylläpitäjät kohtaavat nykypäivänä suurimpana haasteena täyttää tiedonsiirtoverkolle asetetut vaatimukset, kun yrityksissä järjestelmien ylläpitoon ja kehitykseen tarkoitettua budjettia leikataan jatkuvasti. Ylläpitäjät joutuvat käyttämään laite-tason hallintaa ja manuaalisia prosesseja verkon ylläpitämiseen. (ONF, 2012, s. 5)

SDN-arkkitehtuurissa hallinta-taso eriyttämällä, mahdollistetaan arkkitehtuurin alempien kerroksien skaalautuvuus ja muokattavuus sovelluksille ja tietoliikenneverkon palveluille (ks. Kuva 1). (ONF, 2012, s. 7)



Kuva 1. SDN-arkkitehtuuri (ONF, 2012, s. 7)

SDN-arkkitehtuurissa tietoliikenneverkon looginen hallinta on keskitetty SDN-hallintasovellukselle. Hallintasovellus ylläpitää loogista kuvaa koko tiedonsiirtoverkosta. Tuloksena sovellukset ja ohjelmat näkevät tietoliikenneverkon yhtenä suurena kytkimenä. SDN:n avulla yritykset ja operaattorit saavat valmistajasta riippumattoman hallinnan koko tiedonsiirtoverkolle yhdestä loogisesta pisteestä, mikä suuresti helpottaa verkon suunnittelua ja operointia. (ONF, 2012, s. 7)

Tärkeimpänä ominaisuutena tietoliikenneverkkoa voidaan hallinnoida ohjelmallisesti, jolloin ei tarvitse ylläpitää tuhansia riviä konfiguraatiota. Ohjelmallisesti hallittavassa tietoliikenneverkossa muutokset voidaan tehdä ja ottaa käyttöön nopeasti. SDN-arkkitehtuurissa ei tarvitse tehdä usealle laitteelle erikseen muutoksia asetuksiin, muutokset tehdään yhteen loogiseen paikkaan. (ONF, 2012, s. 7)

SDN-arkkitehtuuri tukee API-rajapintaa. API-rajapinnan avulla voidaan implementoida yleisiä tietoliikenneverkon palveluita, kuten reitittämistä, multicast-tiedonsiirtoa, tietoturvakomponentteja, kaistan hallintaa, liikenteen muokkaamista, QoS:ia, optimoida prosessointia ja tallennustilan käyttöä. SDN-arkkitehtuuria voidaan soveltaa niin langalliseen kuin myös langattomaan tiedonsiirtoon. (ONF, 2012, s. 8)

3.3 **Automatisoinnin hyödyt**

Manuaalisesti toteutetussa tietoliikenneverkon päätelaitteen konfiguroinnissa jokaiselle päätelaitteelle operaatiot tehtäisiin jokaiselle laitteelle yksi kerrallaan. Esimerkiksi reitittimelle konfiguroinnin luominen alusta asti manuaalisesti olisi erittäin työlästä ja aikaa vievää sekä manuaalisesti konfiguroidessa virheiden todennäköisyys kasvaa mitä enemmän konfigurointirivejä päätelaitteelle syötetään.

Automatisoidussa ympäristössä tämä päätelaitteiden konfigurointi voidaan suorittaa monelle laitteelle samaan aikaan ilman manuaalisia toimenpiteitä.

Skripteillä eli komentosarjoilla voidaan hallitusti muokata useita tietoliikenneverkon laitteita samanaikaisesti tekemällä komentoja jollain komentosarjakielellä eli skriptikielellä. Komennoilla voidaan tietoverkkolaitteille asettaa tai hakea tietoja. Skriptaustavat, käytetystä skriptikielestä riippumatta, ovat yleensä kahta eri päätyyppiä: verkkolaitteella tapahtuva skriptaaminen ja palvelimella tapahtuva skriptaaminen. Periaatteellisena erona näissä tavoissa on se, missä järjestelmässä skriptikoodi tulkitaan. (Lappalainen, 2010)

Verkkolaitteella tapahtuvassa skriptauksessa skriptit ovat yleensä tapahtumaohjattuja, ts. ne toteutetaan vasta jonkin tapahtuman toteutuessa paikallisella laitteella. Nämä tapahtumat voivat olla esim. muutokset reititystaulussa, tai jokin ajastettu toiminto. Palvelinskriptauksessa palvelin voi saada usealta verkonlaitteelta esim. SNMP-trap viestin, jolloin palvelin viestin saatuaan tekee tarvittavat toiminnot usealle eri verkkolaitteelle. Tämmöisiä toimintoja voi olla viestin raportointi eteenpäin sähköpostilla, tai skriptin syöttäminen verkkolaitteille. (Jones, 2005, s. 1; ks. myös Lappalainen, 2010)

Automatisoinnin etuna on nopea ja luotettava vastaavuus tietoliikenneverkon muutoksiin. Tietoliikenneverkkoon voidaan tehdä nopeasti ja hallitusti muutoksia tietoturvallisesti, kun muutokset tehdään prosessinomaisesti. Prosessinomaisessa muutoksen hallinnassa muutoksia voidaan testata, katselmoida, hyväksyttää ja ajastaa käyttöön. Kun verkkomuutoksille on luotu tietty proseduuri, vähentää se virheellisten muutosten riskiä. Jokaisen prosessin vaiheen voi hyväksyttää eri henkilöllä. Prosessinomaisella tietoverkonhallinnalla voidaan seurata verkkoon tehtyjä muutoksia, josta on hyötyä jälkeenpäin vianselvityksessä. (Jones, 2005, s. 14)

Prosessinomaisella muutoksen hallinnalla on myös ristiriitansa. Esimerkiksi, kun samalle verkkolaitteelle kaksi tai useampi eri henkilö tekee muutoksia voi syntyä laitteen asetuksiin ristiriita. Henkilöt tekevät muutokset verkkolaitteen alkuperäisten asetusten pohjalta, yksi muutoksista hyväksytään ennen muita ja otetaan tuotantoon. Muiden henkilöiden tekemät muutokset ovat ristiriidassa ensimmäiseksi hyväksyttyjen muutoksien

kanssa. Ratkaisuna tähän on prosessiin luodut vaiheet ja ajastettu toteutus. Kun muutokset lähetetään tarkasteltavaksi, muutosten tarkastajan tulee tiedottaa muita henkilöitä, jotka tekevät muutoksia verkkolaitteille. Näin muut henkilöt ovat tietoisia muista muutoksista. Kun prosessi suoritetaan ajastetusti, vähennetään ristiriidan todennäköisyyttä. (Jones, 2005, s. 15)

Johdonmukaisella prosessilla muutokset suunnitellaan, hyväksytään, ja toteutetaan vain kerran. Prosessin etuna on, että kaikki tehdään manuaalisesti vain yhden kerran. Jokaiselle verkkolaitteelle ei tarvitse erikseen tehdä samoja vaiheita. Johdonmukainen toimita parantaa tietoliikenneverkon tietoturvallisuutta, hallittavuutta, sekä sen luotettavuutta. (Jones 2005, s. 15)

Muutostenhallinta auttaa palautumaan ongelmatilanteista. Ongelmatilanteita voi syntyä äkillisestä laiteviasta, sähkökatkosta, tai väärin toteutetusta laitekonfiguraatiosta. Muutosten hallinnassa jokaisesta toimivasta verkkolaitekonfiguraatiosta tallennetaan varmuuskopio. Varmuuskopiota voidaan jälkeenpäin hyödyntää ongelmatilanteista toipumiseen. (Jones 2005, s. 15-16)

Muutosten hallinnassa varmuuskopiointi tulee suorittaa ajastetusti ja myös aina kun laitteelle tapahtuu muutoksia. Tällainen varmuuskopiointi voi olla käyttäjän tekemien muutoksien lisäksi lokitietojen monitorointi, autentikointi liikenne, tai SNMP-viestit. Varmuuskopiointia voidaan tehdä myös silloin kun laitteelle oletetaan tulevan muutoksia, esim. kun käyttäjä kirjautuu verkkolaitteelle sisään. (Jones, 2005, s. 15-16)

Automatisoinnilla voidaan tehokkaasti muuttaa asetuksia useaan tietoverkkolaitteeseen samanaikaisesti. Manuaalisesti toteutettu muutos useaan verkkolaitteeseen on aikaa vievää ja vaivalloista. Jos sataan verkkolaitteeseen tulisi muuttaa laitteen salasana, tulisi ensiksi laskea kuinka paljon menee aikaa muuttaa yhden laitteen salasana, sen jälkeen kertoa se laitteiden määrällä. Tämänlaiseen toimenpiteeseen voi mennä hyvinkin kauan. Automatisoidussa tietoliikenneverkossa muutostenhallinnan avulla toimenpide tarvitsee suorittaa vain kerran. Muutostenhallinnan avulla voidaan määrittää verkkolaitteelle tehtävät toimenpiteet, sekä mille verkkolaitteille toimenpide suoritetaan. (Jones, 2005, s. 19)

Tietoliikenneverkkoa tulee voida tarkastella ja kerätä tietoa verkkolaitteiden eri asetuksista. Automatisoidun tietoliikenneverkon etuna on, että sitä voidaan helposti ja jatkuvasti tarkastella. Manuaalisessa tietoliikenneverkonhallinnassa jokaisesta laitteesta täytyy erikseen hakea verkkolaitteen asetukset ja tarkastella niitä. Manuaalisesti toteutettuna tämä on virhealtista, kun tarkasteltavia verkkolaitteita on useita satoja, virheen todennäköisyys kasvaa. Jotkin verkkolaitteet voivat unohtua ja kaikkia asetuksia ei

muista tarkastella, kun verkkolaitteen asetuksista voi tulla useita satoja rivejä tekstiä. Automatisoinnin avulla verkkolaitteista voidaan tarkastella vain sitä osaa asetuksista, joihin on tarvetta. (Jones, 2005, s. 19)

4 AUTOMATISOINTI TEKNIIKAT

4.1 Ansible

Ansible on automatisointityökalu, joka soveltuu moniin ICT -alan tarpeisiin. Sen avulla voidaan konfiguroida järjestelmiä, julkaista ohjelmistoja ja orkestroida helposti monimutkaisiakin palvelinratkaisuja, kuten jatkuvaa julkaisua tai ohjelmistojen päivityksiä ilman, että palvelua ajetaan päivityksen ajaksi pois toiminnasta. Ansible on moduuli pohjainen ohjelmisto, joka tarkoittaa, että kaikki suoritettavat konfiguraatiotoimenpiteet toteutetaan yksittäisten kirjastojen kautta. (Hochstein, 2015, s. 2)

Ansible on idempotenssi asetustenhallintatyökalu. Tämä tarkoittaa, että Ansiblen suorittamien komentojen jälkeen näiden tila on aina sama huolimatta siitä, ajetaanko asetus-skripti ensimäistä tai viidettä kertaa. (Hochstein, 2015, s. 6)

Ansiblen keskiössä ovat ns. käsikirjat (playbooks). Käsikirjoissa määritellään, mitkä järjestelmät tai laitteet sen avulla konfiguroidaan ja niissä myös listataan tehtävät (tasks), jotka konfiguroinnissa ajetaan. Käsikirjat kirjoitetaan ja määritellään YAML -syntaksia käyttämällä.

Nämä YAML -merkintäkielellä kirjoitetut Ansible -käsikirjat ovat käytännössä niiden välivaiheiden suoritettavat toimenpiteet, joita määriteltävä tehtävä tarvitsee. Jokainen näistä välivaiheista suoritetaan jonkin tietyn Ansible-modulin avulla. Ansiblessa on kattava valikoima ydin moduuleja (core-modules), jotka sisältävät kaikki perustoiminnot. Lisäksi Ansiblea voidaan laajentaa omilla itse kirjoitetuilla moduuleilla.

YAML on helposti luettava merkintäkieli, jota käytetään erityisesti konfiguraatiotiedostojen määrityksissä. YAML -merkintäkielessä on minimalistinen syntaksi, mikä tekee siitä helposti käytettävän ja luettavan. YAML -merkintäkieli käyttää samoja ohjelmointisyntakseja Python -ohjelmointikielen kanssa, kuten funktioiden sisennystä sekä hakasulkeita listoille ja kaarisulkeita sanakirjoille.

4.2 Puppet

Puppet on avoimen lähdekoodin konfiguraation hallinta -työkalu. Sen kehitti Luke Kanies vuonna 2005 ja se on kirjoitettu C++ ja Clojure -ohjelmoin-

tikielillä alkaen versiosta 4.0. Vanhemmat versiot on kirjoitettu Ruby -ohjelmointikielellä. Puppet on tällä hetkellä yksi vanhimmista automaatioon tarkoitetuista ohjelmistoista.

Puppet konfiguraatiot eli moduulit määritellään käyttämällä Puppetin omaa määrittelykieltä DSL:ää (eng. domain specific language).
Puppet -ympäristön toiminta vaatii, että se on asennettu kaikkiin hallittaviin järjestelmiin, joka puolestaan edellyttää Ruby -ympäristöä jos käytössä on vanhempi versio kuin 4.0 sekä Puppet -agentin. Nämä lisäävät keskusmuistin käyttöä järjestelmässä merkittävästi, jos käytössä on pienitehoinen virtuaalikone. Laitteiden välinen verkotus täytyy myös tehdä niin, että laitteet löytävät toisensa isäntänimen (eng. hostname) perusteella. Jos pilvipalveluntarjoaja ei aseta laitteille täydellistä verkkonimeä (eng. fully qualified domain name, FQDN), on käytettävä omaa DNS-palvelinta (eng. domain name system).

Puppet voi olla sekä push että pull -tyyppinen automaatiotyökalu. Oletusarvoisesti Puppet -agentit kysyvät päälaitteelta (eng. master) konfiguraatiotietoja 30 minuutin välein. Jos master-laitteen lähettämä konfiguraatio eroaa viimeksi ajetusta, Puppet -agent ajaa uuden konfiguraation. On myös mahdollista käskeä Puppet -agentteja hakemaan konfiguraatio pyynnöstä, mutta se vaatii mcollective ja activemqn -moduleiden asentamisen, jotka lisäävät entisestään Puppetin korkeahkoja laiteresurssivaatimuksia.

4.3 **NAPALM**

NAPALM (eng. network automation and programmability abstraction layer with multivendor support) on avoimen lähdekoodin Python -kirjasto, joka on erityisesti tarkoitettu tietoliikenneverkon laitteiden konfiguroinnin automatisointiin. NAPALM:ssa on yhteinäinen rajapinta eli API (eng. application program interface) monille verkkolaitejärjestelmille kuten, Cisco IOS ja IOS-XR, Juniper JunOS ja Arista EOS.

Esimerkiksi konfiguraatio tietojen hakuun verkkolaitteelta voidaan käyttää funktiota ”get_bgp_neighbors()”, joka on sama komento kaikille verkkolaitteille, vaikka niissä kaikissa olisi eri käyttöjärjestelmä.

NAPALM:n voi asentaa helposti Pythonin paketinhallinnasta komennolla ”pip install napalm”, tällä tavalla asennettuna siinä voidaan käyttää myös itse kirjoitettuja skriptejä ja moduuleita.

4.4 **TCL**

TCL (eng. Tool Command Language) ohjelmointikielen on keksinyt John K. Ousterhout 1980 luvun loppupuolella Kalifornian yliopistossa Berkleyssä.

TCL on dynaaminen ohjelmointikieli, jota voidaan käyttää skriptien luontiin, tulkkikielenä ja C-kirjastona. TCL auttaa hallitsemaan ja komentamaan muita ajettavia ohjelmia ja toimintoja ohjelmassa. TCL on tulkattu ohjelmointikieli. Tulkatun ohjelmointikielen etuna käännettyyn ohjelmointikieleen on sen nopea kehitysprosessi; skriptiin voidaan tehdä muutoksia nopeasti ja suorittaa niitä nähdäkseen muutoksen tuomat vaikutukset. Etuna on myös skriptien luominen tekstimuodossa. Tulkkikielen haittapuolena on sen suorituskyky. Suorituskyky riippuu ajettavan alustan käyttöjärjestelmästä, prosessorista ja ohjelmointikielestä. TCL-skripti täytyy ensiksi tulkata, ennen kuin se suoritetaan. Toinen haittapuoli on skriptikoodin piilottaminen. Koska skriptit tehdään tekstimuodossa, voi kuka tahansa, joka saa skriptin käyttöönsä, muokata ja kopioida sitä. Suorituskyvyn lisäksi tulkattavissa ohjelmointikielissä on haittapuolena niiden muistin vaatimus. Koko skripti, sen tulkattu muoto ja muuttujat pidetään välimuistissa. (Blair, Durai, Lautman, 2010, 1)

TCL -skriptaamisen edut ovat tietojen tulostaminen ja hakeminen muista laitteista, käyttöliittymistä ja tietokannoista. Skripteillä voidaan myös automatisoida monimutkaisia tehtäviä. Informaatioita voidaan muokata erilaisilla kokonaisluvuilla ja muuttujilla. TCL on myös helposti opittava skripti kieli. (Blair, Durai, Lautman, 2010, 1)

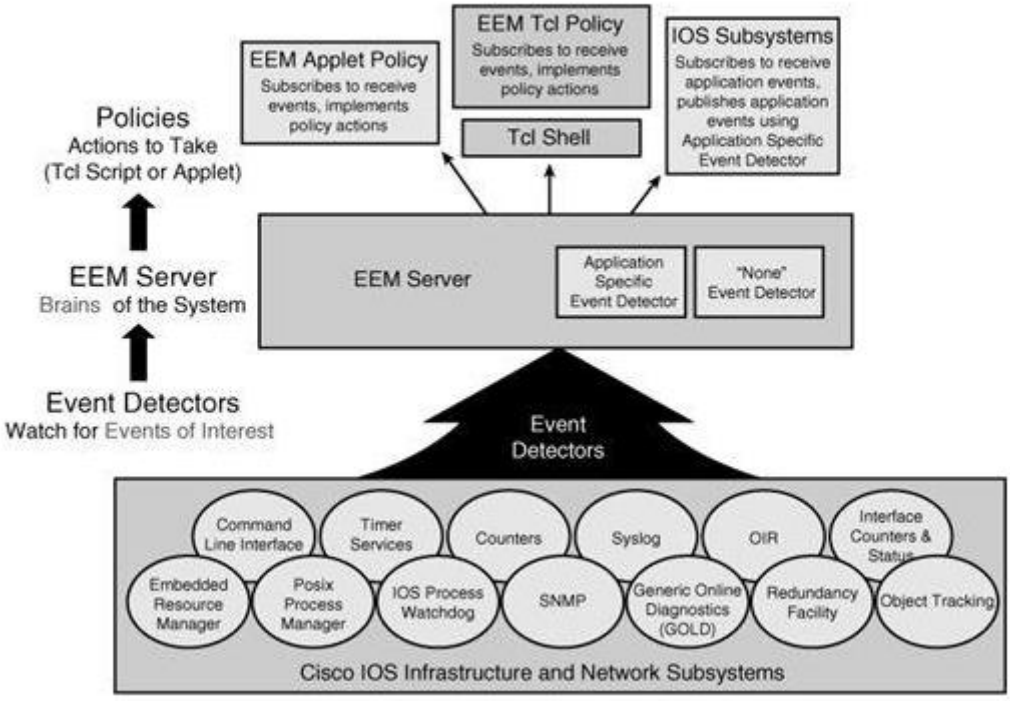
TCL -ohjelmointikieleen kuuluu myös useita eri komponentteja, joista tunnetuin on TK (eng. Tool Kit) -komponentti. TK-komponentin avulla ohjelmalle voidaan luoda graafinen käyttöliittymä. Komponentilla voidaan luoda ikkunoita, painikkeita, tekstilaatikoita jne. TCL -ohjelmointikieltä käytetään yleisesti ohjelmien testaamiseen, automatisointiin, web-käyttöliittymissä, työpöytäohjelmissa, tietokannoissa ja sulautetussa ympäristöissä. (Blair, Durai, Lautman, 2010, 1)

TCL -ohjelmointikieli on saavuttanut suosiota helppona ohjelmointikielenä, jota voi ajaa lähes kaikilla alustoilla. Se eroaa muista skriptikielistä siinä, että sen voi helposti sulauttaa muihin sovelluksiin. Lisäksi TCL on ilmainen ja TCL -yhteisön ylläpitämä. (Blair, Durai, Lautman, 2010, 1)

Ciscon IOS -käyttöjärjestelmissä TCL tuki esitettiin ensimmäisen kerran versiossa 12.3(2)T ja 12.2(25) S. TCL tuki tuli osaksi Cisco IOS -käyttöjärjestelmää Catalyst 6500 -sarjan modulaarisissa kytkimissä versiossa 12.2(18) SX4. IOS -käyttöjärjestelmistä TCL -skripti voidaan ajaa omasta käyttöliittymästä. Käyttöliittymään pääsee käsiksi komennolla ”tclsh”, komennon voi suorittaa vain EXEC -tilassa. Cisco IOS -käyttöjärjestelmässä useat aliohjelmat tukevat TCL -skriptejä, tällaisia aliohjelmia ovat mm. ESM, EMM, IVR ja EEM. (Blair, Durai, Lautman, 2010, 1)

Alla olevassa kuvassa esitelty CISCO IOS -käyttöjärjestelmän EEM -aliohjelma (embedded event manager), joka mahdollistaa TCL -skriptien suorittamisen suoraan CISCO:n tietoverkkolaitteissa.

Figure 1-1. EEM's Relationship with Other Functions



Kuva 2. CISCO IOS -aliohjelma EEM:n toiminta. (Blair, Durai, Lautman, 2010, 1)

5 OPINNÄYTETYÖN TOTEUTUS

5.1 Yleistä

Opinnäytetyön toteutusosiossa keskitytään tarkemmin itse ohjelmakoodin ja tietokannan rakenteeseen sekä ohjelmointiin. Osiossa esitetään syitä, että miksi joku operaatio on tehty ohjelmistoon sekä mitä tällä operaatiolla saavutetaan. Ohjelmakoodin salassapidon takia itse lähdekoodia eikä tietokantaa sellaisenaan voi esittää.

Ohjelmiston toiminta perustuu täysin siihen, että päätelaitteelle on tehty jo aloituskonfiguraatio ja se on kytketty tietoliikenneverkkoon, jotta ohjelmiston ensimmäinen testausoperaatio saisi yhteyden päätelaitteeseen. Aloituskonfiguraation voi käytännössä tehdä missä sijainnissa vain mutta yleensä se tehdään asiakkaan tiloissa käyttöönoton yhteydessä. Aloituskonfiguraatiossa päätelaitteelle määritellään hallinta IP-osoite, staattinen reitti Elisan runkoverkkoon, jotta laite olisi tavoitettavissa hallintaverkon kautta sekä SNMP -salasanat. Aloituskonfiguraation suorittaa yleensä kenttäasentaja mutta se voidaan suorittaa myös esimerkiksi päätelaitteen lähtöpisteessä ja asiakkaan toimitiloissa vain tehdä paikalleen asennus ja kytkennät.

5.2 Ohjelmiston tietokanta

Itse opinnäytetyön ohjelmistoa varten suunniteltiin ja toteutettiin tietokanta päätelaitteiden tallennusta ja niille tehtäviä operaatioita varten. Tietokantaohjelmistoksi valikoitui MariaDB, kuten opinnäytetyön alussa kerrotaan.

Ohjelmistoon suunnittelut testit vaativat, että päätelaitteet lisätään tietokantaan, jotta niille suoritettavia operaatioita voidaan tarkastella ja analysoida sekä tarvittaessa manuaalisesti käsitellä niitä.

Päätelaite tietokantaan luotiin taulu, johon päätelaitteet lisättäisiin aina kun uusi päätelaite on valmis viimeistelyprosessiin. Tauluun luotiin sarakkeet liittymän tunnisteelle, hallinta IP-osoitteelle, päätelaitevalmistajalle sekä aikaleima (eng. timestamp) tietueelle. Jokaiselle testausoperaatiolle luotiin myös sarakkeet, joissa ilmoitetaan, onko testausoperaatio onnistunut vai onko se epäonnistunut sekä myös aikaleima-tietue. Lopuksi lisättiin vielä lopullisen onnistumisen ilmoittava sarake, joka niin ikään ilmoittaa onnistumisen joko arvolla onnistui tai arvolla epäonnistui.

Ohjelmiston ensimmäinen testausoperaatio oli päätelaitteen tavoitettavuuden tarkistus eli onko päätelaite tavoitettavissa operaattorin, tässä tapauksessa Elisan, hallintaverkosta käsin. Tätä operaatiota varten tietokan-

taan lisättiin edellä mainitut sarakkeet eli sarake, joka ilmoittaa onnistumisen sekä aikaleiman, johon päivitetään onnistumisen päivämäärä ja kellonaika.

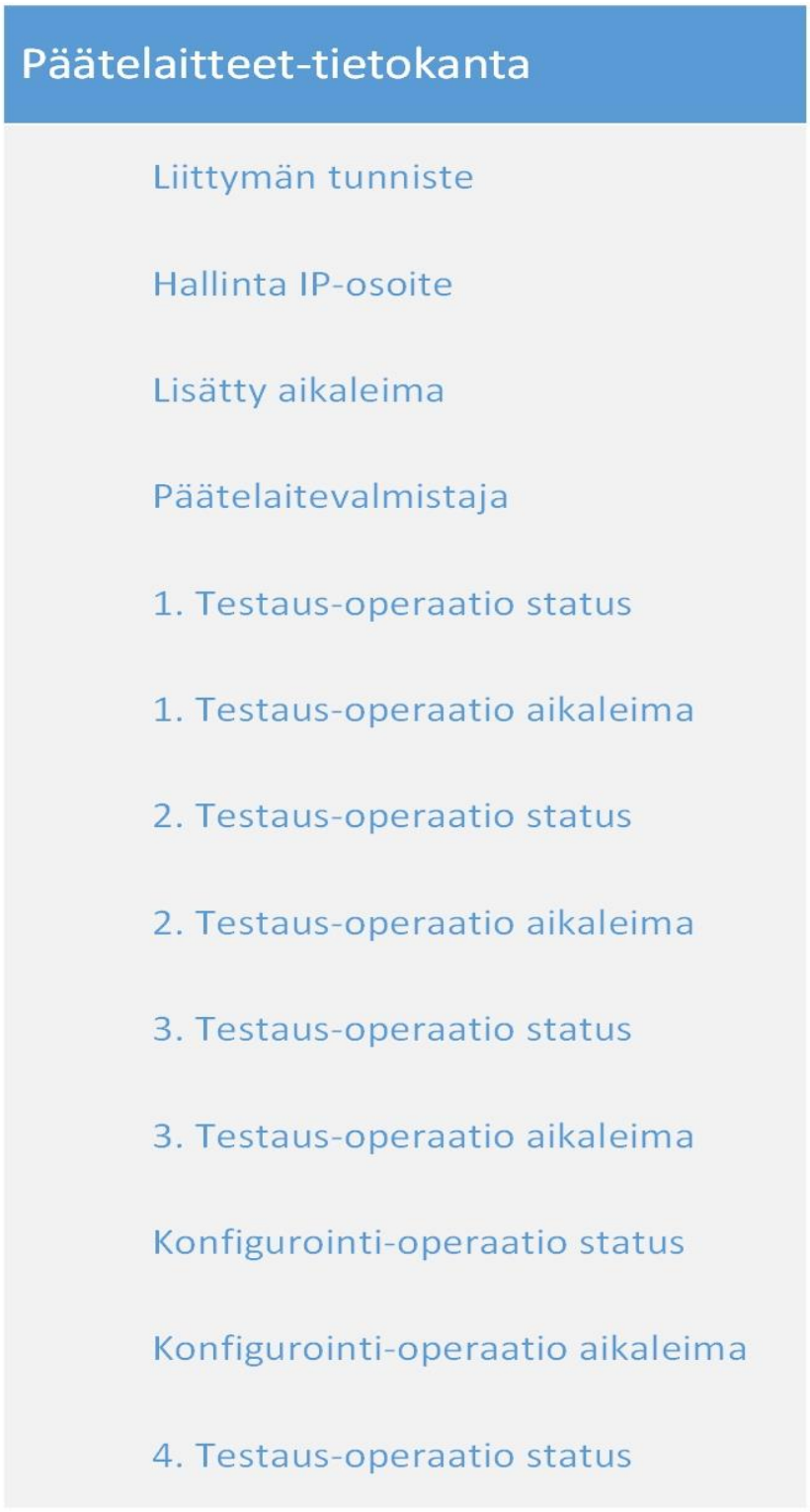
Ohjelmiston toinen testausoperaatio oli yritysverkkoliittymän nopeusmittauksen tarkistus. Tässä nopeusmittauksessa mitattiin yritysasiakkaan päätelaitteen ja Elisan runkoverkon välinen datan siirtonopeus niin lähetysnopeus kuin latausnopeuskin. Testausoperaatiossa huomioitavin asia oli tarkka aika, jolloin nopeusmittaus on suoritettu kyseiselle päätelaitteelle. Ohjelmisto valikoi nopeusmittaustulokset erinäisestä nopeusmittaustietokannasta yhden tunnin sisällä siitä ajankohdasta, kun valintafunktio suoritettiin. Tällä voitiin varmistaa, että nopeusmittaustulos operaation kohteena olevalle päätelaitteelle on aina uusin eikä koko nopeusmittaus tietokannan taulua tarvitse tallentaa ohjelmiston muuttujaan (eng. variable). Ohjelmiston tietokantaan lisätyt sarakkeet tälle nopeusmittaus-operaatiolle ovat samanlaiset kuin edellisellekin eli sarake, joka ilmoittaa onnistumisen arvoilla kyllä tai ei ja aikaleima-tietue, joka lisätään heti, jos oikea nopeusmittaustulos löytyy oikealle päätelaitteelle ja oikean aikahaarukan sisällä.

Ohjelmiston kolmatta testausoperaatiota varten päätelaite tietokantaan lisättiin niin ikään sarake onnistumista varten sekä aikaleima-tietue ilmoittamaan milloin testausoperaatio on onnistunut. Kolmas testausoperaatio oli päätelaitteen valmistajan ja mallin tarkistaminen. Testausoperaatiossa tarkistettiin, että vastaako tietylle päätelaitteelle tehty konfiguraatio testausoperaation kohteena olevaa päätelaitetta merkiltään ja malliltaan. Toisin sanoen testausoperaatio tarkistaa onko testausoperaatiossa oleva päätelaite merkiltään ja malliltaan esim. CISCO 892FSP ja onko tälle päätelaitteelle lopuksi ajettava konfiguraatiotiedosto samalle merkille ja mallille.

Ohjelmiston neljäs eli toiseksi viimeinen vaihe oli itse päätelaitteen konfigurointi. Tätä varten tietokannan suunnittelu noudatti samoja periaatteita kuin tätä vaihetta edeltävissä testausoperaatioissakin. Luotiin sarake päätelaite tietokantaan ilmoittamaan operaation onnistumista tai epäonnistumista sekä sarake aikaleima-tietueelle ilmoittamaan operaation mahdollisen onnistumisen aika.

Ohjelmiston operaatioiden viimeinen vaihe oli konfiguroidun päätelaitteen tavoitettavuus sen jälkeen, kun konfigurointi on ajettu onnistuneesti ensin läpi. Tätä varten tietokantaan lisättiin myös sarake ilmoittamaan koko testaus- ja konfiguraatioprosessin onnistumista. Tämä testausoperaatio ilmoitettiin myös onnistuneeksi tai epäonnistuneeksi arvoilla onnistui tai epäonnistui. Muista testausoperaatio sarakkeista poiketen tälle operaatiolle suunniteltiin ensin, että aikaleima sarake jätettäisiin pois tietokannasta mutta mietinnän jälkeen tultiin tulokseen, että se olisi järkevää lisätä tietokantaan myöhempiä analyyseja varten.

Päätelaite tietokantaan toteutettiin vain yksi taulu, johon kaikki päätelaitteet lisättäisiin. Tietokantaan ei tämän takia tarvinnut tehdä relaatioita ollenkaan. Alla karkea havainnekuva päätelaitteet -tietokannan taulusta (ks. Kuva 3).



Kuva 3. Päätelaitteet -tietokannan taulun havainnekuva

5.3 **Opinnäytetyön ohjelmiston toteutus**

Itse opinnäytetyön ohjelmisto ohjelmoitiin Python -ohjelmointikielellä. Python tulkista käytettiin versiota 2.7 (2.7.5), koska versio 2.7 oli yhteensopiva muiden ohjelmiston käyttämien järjestelmien kanssa. Myöskin kummallakin testipalvelimella oleva versio Python -tulkista oli 2.7.

Ohjelmistoa alettiin ohjelmoimaan operaatio kerrallaan ja jokaisesta operaatiosta tehtiin oma luokka, jonka sisälle varsinaiset operaation suorittavat funktiot ohjelmoitiin. Tämä sen takia, että lähdekoodista saatiin helpommin laajennettava ja skaalautuva mahdollisille uusille operaatioille ja funktioille. Koodissa olevia luokkia on mahdollista myös käyttää toisissa Python -ohjelmissa käyttämällä ns. import -komentoa koodin alussa, jossa lisätään koodiin toisesta koodista jo olemassa oleva luokka tai funktio. Operaatioiden ohjelmoiminen omiin luokkiin tekee koodista myös koodista helppolukuisen.

Testausoperaatioita varten tarvittiin myös rajapinta ohjelmiston käyttämiä tietokantoja varten. Python -koodiin ladattava (eng. import) rajapinta -moduuli on tällä hetkellä vain saatavilla MySQL -tietokannan yhdistämiseen mutta samaa moduulia käytetään myös MariaDB -tietokannan yhdistämiseen. Ohjelmistossa käytettävä nopeustesti tietokanta oli MySQL -ohjelmistolla toteutettu ja päätelaite tietokanta MariaDB -ohjelmistolla, joten tämä moduuli oli yhteensopiva kummankin kanssa.

Tietokantojen operoimista varten koodiin rakennettiin oma luokka, johon tehtiin jokaista tietokannan haku, lisäys, päivitys sekä poisto -operaatiota varten funktiot. Nämä funktiot sisälsivät tietokantaan tehtävät SQL -operaatiolauseet (eng. SQL -clause), joita jokainen testausoperaatio käytti.

Huomioitavaa ohjelmiston rakenteessa on myös se, että edellisen testausoperaation täytyy olla suoritettuna, jotta seuraava testausoperaatio voisi tulla suoritetuksi.

Ohjelmisto konfiguroitiin toimimaan automaattisesti Elisan testipalvelimella Linuxin cron -ajastustoiminnon avulla. Cron on Linux -käyttöjärjestelmässä natiivisti asennettuna ja sitä käytetään Crontab -ohjelman kautta. Crontab -ohjelma ohjaa crond -ajastuspalvelua, joka ajaa annetut komennot taustalla ja tarkistaa annetun ajan välein, jos komentoja pitää suorittaa. Konfigurointiohjelmisto konfiguroitiin aluksi toimimaan 1 minuutin välein palvelimella.

Ohjelmistoon lisättiin myös tietokantaan lisäys -funktio, jotta yritysverkon provisioinnin yhteydessä päätelaite voidaan lisätä päätelaite-tietokantaan. Lisäys -funktiossa Python -funktiolle annetaan liittymätunniste sekä hallinta IP-osoite, jolloin päätelaite luodaan päätelaite-tietokantaan INSERT-lauseella. Päätelaite saa nämä annetut arvot sekä ajankohdan, milloin laite on lisätty tietokantaan.

Tietokannasta poisto -funktio lisättiin myös ohjelmistoon varmistuksena, että päätelaite voidaan myös poistaa manuaalisesti tietokannasta. Poisto -funktio tehtiin käyttämällä DELETE -lausetta, jolle annettiin arvona liittymätunniste.

5.3.1 Ensimmäinen testausoperaatio

Ensimmäisessä testausoperaatiossa päätelaite tietokannasta haettiin tavanomaisella SQL-SELECT -lauseella kaikki päätelaitteet, joissa tavoitettavuus oli vielä epäonnistunut -tilassa. Jos päätelaitteella oli tavoitettavuus sarakkeessa epäonnistunut -tila, niin se tarkoitti, että päätelaite oli juuri hetki sitten lisätty tietokantaan tai se ei ole ollut tavoitettavissa siitä asti, kun se on lisätty tietokantaan. Tietokannan hakemille laitteille tehtiin seuraavaksi ping -testi eli jokaiselle päätelaitteelle lähetettiin ping -sanoma (eng. icmp echo request, ping), joka testaa onko päätelaitteen IP-osoite tietoliikenneverkossa tavoitettavissa. Kun ping -testi oli suoritettu päätelaitteille niin niille laitteille, jotka lähettivät ping -vastauksen (eng. ping -reply) palvelimelle, lisättiin tietokantaan tavoitettavuus sarakkeeseen onnistunut -status sekä aikaleima tälle statukselle.

5.3.2 Toinen testausoperaatio

Toisessa testausoperaatiossa haettiin SQL-SELECT -lauseella erillisestä nopeustesti tietokannasta kaikki yhden tunnin sisällä olevat tulokset ja näistä tuloksista tallennettiin muuttujaan kaikkien liittymänumerot. Näiden liittymänumeroiden perusteella päivitettiin päätelaite tietokantaan SQL-UPDATE -lauseella niiden päätelaitteiden nopeustesti -status onnistuneeksi, jotka täsmäävät muuttujassa oleviin liittymänumeroihin ja joilla on tavoitettavuus testi onnistunut.

5.3.3 Kolmas testausoperaatio

Kolmannessa testausoperaatiossa tarkistettiin päätelaitteen valmistaja ja malli. Tarkistus suoritettiin vertaamalla päätelaitteen konfiguraatiotiedoston valmistaja ja malli -kenttiä päätelaitteen palauttamaan tietoon. Tiedon haku päätelaitteelta suoritettiin käyttämällä SNMP -get viestiä (eng. simple network management protocol), johon valittiin tarkoituksen mukainen OID (eng. object identifier) eli tunniste, joka määrittää mitä tietoa laitteelta haetaan. Eri laitemerkit ja -mallit käyttävät eri OID -tunnisteita, joten tähän operaatioon käytettävä OID määritettiin samasta päätelaite konfiguraatiotiedostosta, vaikka ei voitu olla täysin varmoja siitä onko päätelaitteen valmistaja juuri tämän OID:n mukainen.

Kun päätelaitteet olivat tarkistettu, niiden tiedot päivitettiin päätelaite tietokantaan. Mikäli SNMP -kyselyn palauttama tieto päätelaitteen valmistajasta täsmäsi konfiguraatiotiedoston saman arvon kanssa, niin päätelaite tietokantaan merkittiin päätelaitteen valmistajan kohdalle tämä oikea tieto. Myös testausoperaation mukaisesti, päätelaitteen testisarake päivitettiin, joko arvoon onnistui tai epäonnistui. Samassa yhteydessä päivitettiin myös testin aikaleima sarake, mikäli testi onnistui tälle kyseisellä päätelaitteella.

5.3.4 Konfigurointi operaatio

Ohjelmiston neljäs operaatio eli varsinainen päätelaitteen konfigurointi tapahtui käyttämällä erillistä Pythonin netmiko -kirjastoa. Netmiko on SSH -yhteyden (eng. secure shell) kautta toimiva, tietoverkkolaitteiden hallintaan ja konfigurointiin tarkoitettu Python kirjasto. Se mahdollistaa automatisoidut operaatiot tietoverkkolaitteella kirjautumalla ensin laitteelle ja sen jälkeen suorittamalla ohjelmoidut komennot laitteella. Netmiko -kirjaston funktioita käytettiin konfiguraatio-operaatiossa ensin kirjautumaan päätelaitteelle, jonka jälkeen päätelaitteella suoritettiin komennot, jotka kopioivat valmiin päätelaitekonfiguraation tftp -palvelimelta (eng. trivial file transfer protocol), tallensivat haetun konfiguraation aloituskonfiguraatioksi ja lopuksi käynnistivät laitteen uudelleen.

Konfiguraatio-operaation päätteeksi kaikille operaation kohteena oleville päätelaitteille lähetettiin ping -kysely, jotta pystyttiin tarkistamaan tässä vaiheessa, oliko päätelaite onnistuneesti konfiguroitu ja uudelleenkäynnistys aloitettu. Tässä tapauksessa ping -kyselyn tulos eroaa ensimmäisestä testistä, koska ping -kyselyyn ei odoteta vastausta niiltä laitteilta, jotka ovat onnistuneesti hakeneet konfigurointitiedoston ja aloittaneet uudelleenkäynnistyksen. Eli toisin sanoen niistä päätelaitteista, jotka eivät vastaa ping -kyselyyn testin päätteeksi, koostetaan lista muuttujaan, joka taas lähetetään edelleen tietokanta -luokan UPDATE -funktiolle. UPDATE -funktio päivittää päätelaite tietokantaan listan perusteella, niille päätelaitteille konfiguroinnin suoritetuksi sekä aikaleiman, jotka täsmäävät listan IP-osoitteisiin.

5.3.5 Neljäs testausoperaatio

Viides ja myös viimeinen testausoperaatio tässä versiossa ohjelmistoa oli konfiguroidun päätelaitteen tavoitettavuus. Testin tarkoituksena oli tehdä ping -kysely päätelaitteelle, jonka konfiguroinnista on kulunut vähintään 10 minuuttia. Kulunut aika valittiin sen perusteella, että yritysverkko tason päätelaitteissa uudelleenkäynnistys voi viedä huomattavan paljon aikaa ja 10 minuutin aika tähän testiin katsottiin sopivaksi. Testin tarkoituksena oli

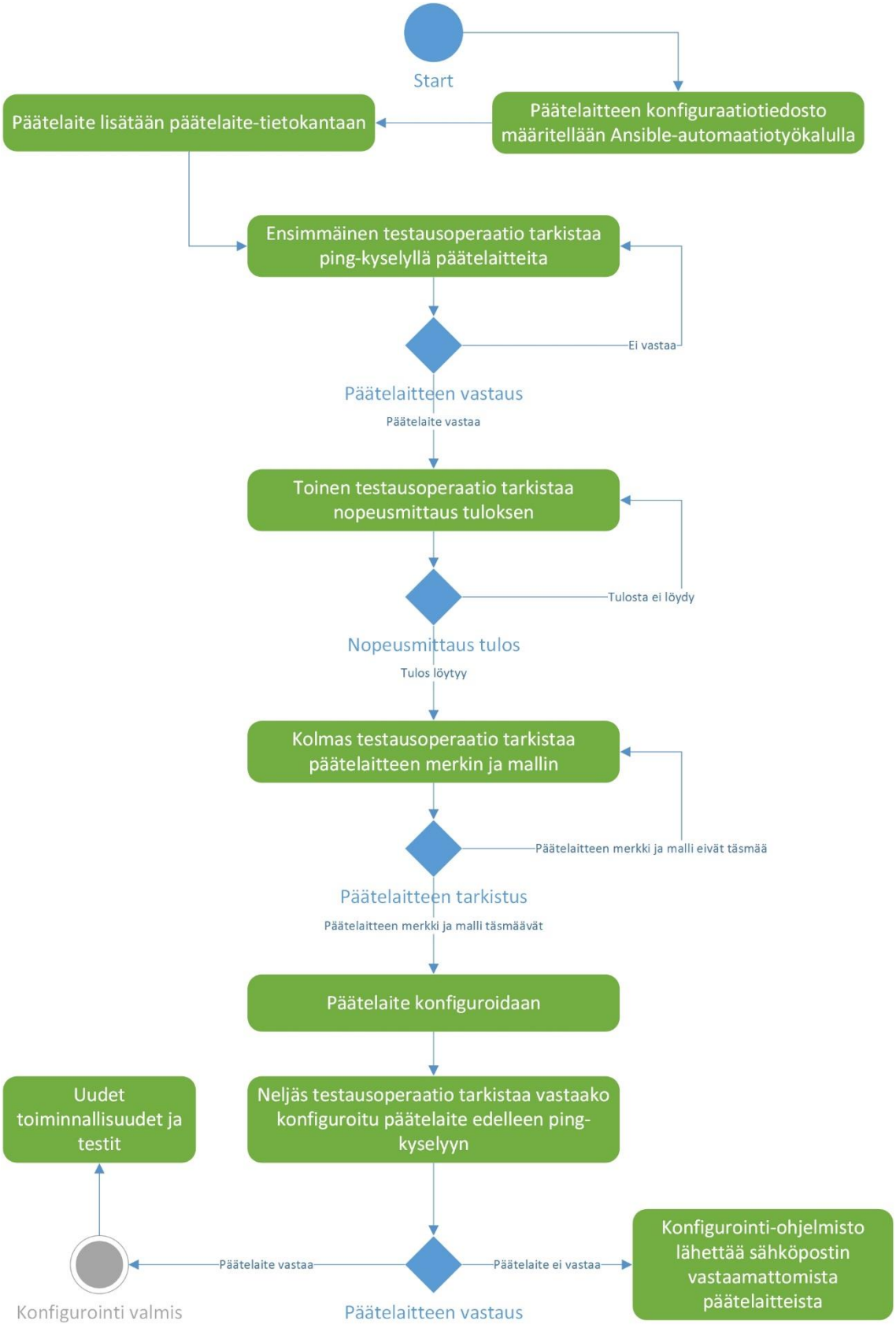
tarkistaa, että onko päätelaite vielä tavoitettavissa konfiguraatio-operaation jälkeenkin.

Jos päätelaite edelleenkin vastaa ping -kyselyyn, niin kaikista laitteista, jotka vastasivat ping -kyselyyn, koostetaan lista muuttujaan, joka lähetetään eteenpäin tietokanta -luokan tämän testin UPDATE -funktiolle. UPDATE -funktio taas päivittää päätelaite-tietokantaan tämän viimeisen testin kohdalle operaation suoritetuksi sekä aikaleiman, niille päätelaitteille, joiden IP-osoite vastaa muuttujassa olevia IP-osoitteita.

Mikäli viidennen eli viimeisen testin yhteydessä ilmenee päätelaitteita, jotka eivät enää vastaa ping -kyselyyn niin näistä laitteista koostetaan myös lista, jonka ohjelmisto lähettää sähköpostina ennalta määritettyyn yhteiskäyttö sähköpostilaatikkoon. Tätä listaa hyödyntäen voidaan manuaalisesti tarkistaa vastaamattomien päätelaitteiden tilanne sekä palauttaa laitteet takaisin konfiguraatioprosessiin tai mahdollisesti manuaalisesti konfiguroida nämä. Sähköpostin lähettämiseen käytettiin Python -tulkista natiivisti löytyvää smtplib -import modulia.

Konfigurointi-ohjelmiston toiminta UML Activity -kaaviolla esitettynä (ks. kuva 4).

Kuva 4. Konfigurointi-ohjelmiston UML Activity -kaavio



6 TULOKSET

Ohjelmiston testauksessa käytettiin neljää eri mallin ja kahden eri valmistajan päätelaitetta. Valmistajat olivat Cisco ja Huawei. Yksi päätelaitteista oli WAN-liitännältään (eng. wide area network) LTE -tyyppinen (eng. long-term evolution) eli pelkästään mobiiliverkon kautta WAN-verkkoon yhteydessä oleva päätelaite. Päätelaitteen WAN-liitäntä ei vaikuttanut ohjelmiston toimintaan millään tavalla.

Testilaitteet:

HUAWEI AR161EW
CISCO C3925
CISCO C899 ETH LTE
CISCO C897

Testauksessa konfigurointiohjelmisto asetettiin automaattisesti toimimaan Elisan Linux -testipalvelimella 1 minuutin välein. Yksi minuutti katsottiin sopivaksi ajaksi, jotta ohjelmisto ei joutuisi kerralla konfiguroimaan montaa laitetta eikä kenttäasentaja asiakkaan toimitiloissa joutuisi turhaan odottelemaan ohjelmiston toiminnan alkamista. Aikaväliä voidaan joutua muuttamaan, kun ohjelmistoa päästään testaamaan isommalla päätelaitemäärällä mutta tämän opinnäytetyön aikana sitä ei päästy vielä toteuttamaan.

Viimeisimmissä testeissä konfigurointiohjelmiston toimiessa automaattisesti Elisan testipalvelimella, päätelaite tietokantaan lisättiin manuaalisesti testilaitteita alkutiedoilla liittymätunniste, IP-osoite ja aikaleima. Näille testilaitteille tehtiin tämän jälkeen nopeusmittaukset nopeusmittaus tietokantaan, jotta näille laitteille voidaan hakea yhden tunnin sisällä olevat nopeusmittaustulokset.

Kolmatta ja neljättä testausoperaatiota varten testilaitteille tehtiin konfiguraatiotiedostot samalla tavalla, miten ne tehtäisiin normaalisti tuotannossa. Tällä tavalla testilaitteille saatiin suoritettua kolmas testausoperaatio, joka tarkisti päätelaitteen merkin, mallin ja merkin mukaisen OID:n. Luodut konfiguraatiotiedostot mahdollistivat testilaitteiden konfiguroinnin neljännessä testausoperaatiossa. Viimeisin testausoperaatio ei vaatinut manuaalisia toimenpiteitä testausta varten.

Alussa testaukset epäonnistuivat yleensä sen takia, että päätelaite tietokannan päivitykset testilaitteille eivät onnistuneet, minkä takia seuraavalla testillä ei ollut mahdollisuutta onnistua. Virheiden korjausten jälkeen, kun testilaitteille oli luotu lähtökohdat konfiguroinnin onnistumiselle, niin ohjelmisto suoritti onnistuneesti kaikki operaatiot testilaitteille ja konfiguroinnin kokonaisuudessaan.

7 ANALYYSI JA JATKOKEHITYS

Ohjelmiston operaatioiden sekä koko ajan kasvavan päätelaitteiden määrän takia ohjelmistoon täytyisi ohjelmoida enemmän sisäisiä testauksia ja virheen korjauksia sekä virhe ilmoituksia. Ohjelmistoon pitäisi ohjelmoida myös sellaisia virheenhallinta ominaisuuksia, että ohjelmiston toiminta ei lakkaa, jos jonkun päätelaitteen kohdalla tulee virhe. Ohjelmiston testausvaiheessa virhetilanteiden etsimiseen ei käytetty niin paljon aikaa, kun itse varsinaiseen toiminnan testaukseen, joten näitä kaikkia virhetilanteita ei tämän opinnäytetyön aikana ehditty korjaamaan.

Ohjelmistoon olisi hyvä kehittää myös erilaisia ilmoituksia eri tilanteista. Esimerkiksi tilanne, jossa päätelaite on lisätty tietokantaan ja sille on testausoperaatiot suoritettu johonkin testiin asti mutta sen jälkeen päätelaitteen testit eivät ole edenneet, niin ohjelmisto lähettäisi ilmoituksen johonkin kaikista samassa tilanteessa olevista päätelaitteista. Viimeisen testausoperaation sähköpostin lähetys on esimerkki tämänkaltaisista tilanteista.

Tämän opinnäytetyön aikana ohjelmiston kaikkia virheitä ei ehditty korjaamaan eikä päätelaitteiden tilasta kertovia ilmoituksia ehditty ohjelmoimaan. Ohjelmiston kehitys jatkuu Elisa Oyj:ssä.

LÄHTEET

Blair, R. & Durai, A. & Lautman, J. (2010). *Tcl Scripting for Cisco IOS: A Guide to Building and Modifying Tcl Scripts to Automate Network Administration Tasks*. 1. painos. USA: Cisco Press.

Chou, E. (2018). *Mastering Python Networking: Your one-stop solution to using Python for network automation. DevOps, and Test-Driven Development*. 2. Painos. United Kingdom: Packt Publishing Ltd.

Elisa Oyj. (2018). Tietoa Elisasta. Haettu 1.8.2018 osoitteesta http://corporate.elisa.fi/tietoa-elisasta/

Hochstein, L. (2015). *Ansible: Up and Running*. 1. painos. USA: O'Reilly Media Inc.

Jones, D. (2006). *Automating Network Management and Compliance*. 1. painos. USA: Realtimepublishers.com.

Lappalainen, J. (2010) Skriptaus. 12.3.2010. Haettu 9.9.2018 osoitteesta https://wiki.jyu.fi/display/opentvt/Skriptaus

Open Networking Foundation. (2012). Software-Defined Networking: The New Norm for Networks. Haettu 13.4.2012 osoitteesta https://www.opennetworking.org/images/stories/downloads/sdn-resources/white-papers/wp-sdn-newnorm.pdf

O'Connor, T. (2013). *Violent Python: A Cookbook for Hackers, Forensic Analysts, Penetration Testers, and Security Engineers*. USA: Elsevier Inc.

Ratan, A. (2017). *Practical Network Automation: Leverage the power of Python and Ansible to optimize your network*. United Kingdom: Packt Publishing Ltd.

Red Hat Inc. (2018). Red Hat Enterprise Linux. Haettu 23.9.2018 osoitteesta https://www.redhat.com/en/technologies/linux-platforms/enterprise-linux

Rhodes, B & Goerzen, J. (2014). *Foundations of Python Network Programming: The comprehensive guide to network programming and management with Python 3*. USA: Apress Media LLC.

Stackoverflow. (n.d). Questions. Haettu 1.8.2018 osoitteesta https://stackoverflow.com/questions

Sweigart, A. (2015). *Automate the boring stuff with Python: Practical Programming for Total Beginners*. USA: No Starch Press Inc.

Tischer, R & Gooley, J. (2017). *Programming and Automating Cisco Networks: A guide to network programmability and automation in the data center, campus and WAN*. USA: Cisco Press.