TALLINNA TEHNIKAÜLIKOOL
Infotehnoloogia teaduskond

Mikk Aruoja 112229 IASB

VÕRGUSEADMETE KONFIGUREERIMISE AUTOMATISEERIMINE

Bakalaureusetöö

Juhendaja: Rein Paluoja
Emeriitdotsent
Kaasjuhendaja: Randel Raidmets
MSc

Tallinn 2020

Autorideklaratsioon

Kinnitan, et olen koostanud antud lõputöö iseseisvalt ning seda ei ole kellegi teise poolt varem kaitsmisele esitatud. Kõik töö koostamisel kasutatud teiste autorite tööd, olulised seisukohad, kirjandusallikatest ja mujalt pärinevad andmed on töös viidatud.

Autor: Mikk Aruoja

17.05.2020

Annotatsioon

Käesoleva bakalaureusetöö eesmärk on välja selgitada milline lahendus oleks kõige sobilikum, ühe asutuse näitel, kommutaatorite konfigureerimise automatiseerimiseks ning saadud tulemus valideerida. Kõigepealt vaadeldi kolme võimalust automaatseks konfigureerimiseks – nullpuute varustamine ehk ZTP, programmeeritav võrgustus ehk SDN ning konfiguratsioonihalduse instrumendid. Analüüsi tulemusel valiti sobivaks viisiks konfiguratsioonihalduse instrumendid.

Valitud viisi tarkvara leidmiseks analüüsiti nelja populaarsemat instrumenti – Ansible, Chef, Puppet ning Saltstack. Lähtudes püstitatud nõuetest ning analüüsi tulemustest osutus sobivaks tarkvaraks Ansible.

Töö viimases osas tehti prototüüp kasutades valitud lahenduse tarkvara ning valideeriti lahendus kasutades kolme vaikekonfiguratsioonis Cisco kommutaatorit. Töö tulemusena konfigureeriti automaatselt Ansible-ga kommutaatorid ning peale konfigureerimist olid kommutaatorid valmis kasutamiseks ajutises kohtvõrgus.

Väljatöötatud lahendust muudab asutuses ajutise kohtvõrgu loomisel kommutaatorite konfigureerimise efektiivsemaks ning aitab vähendada seadistamisel esinevaid vigu.

Lõputöö on kirjutatud eesti keeles ning sisaldab teksti 24 leheküljel, 6 peatükki, 21 joonist, 1 tabelit.

Abstract

Automation of network devices configuration

The aim of this thesis is to identify which method would be the most suitable, on the example of one institution, for automating the configuration of network switches and to validate the obtained results. First, three options for automatic configuration were analyzed – zero touch provisioning or ZTP, software defined networking or SDN and configuration management tools. As a result of the analysis, configuration management tools were selected as the suitable method.

To find a suitable software for the chosen method, four most popular open-source software were analyzed – Ansible, Chef, Puppet and Saltstack. Based on the given requirements and as a result of the analysis, Ansible turned out to be the most suitable software.

In the last chapter a prototype was created using the software of the chosen solution and was validated using three Cisco switches in their default configuration.

As a result of the chosen software, switches that were used in the prototype, were automatically configured and were ready to be used in a temporary local area network.

The developed solution provides a more efficient way to configure switches for creating a temporary local area network in the institution and helps to reduce configuration errors.

The thesis is in Estonian and contains 24 pages of text, 6 chapters, 21 figures, 1 tables.

Lühendite ja mõistete sõnastik

API	*Application Programming Interface* – reeglid ja vahendid rakendusprogrammi suhtluseks operatsioonisüsteemiga.
ARP	*Address Resolution Protocol* – standardne andmeside protokoll, mis teisendab IP-aadressid füüsilisteks MAC-aadressideks.
Cisco IOS	*Cisco Internetwork Operating System* – Cisco kommutaatorites ja ruuterites kasutatav operatsioonisüsteem.
DHCP	*Dynamic Host Configuration Protocol* – standardprotokoll, mis võimaldab arvutitel automaatselt saada IP-aadressi ja muid tööks TCP/IP-võrgus vajalikke parameetreid.
DNS	*Domain Name system* – TCP/IP-võrgu komponentide nimede hierarhiline süsteem, mis võimaldab teisendada domeeni nimesid IP-aadressideks ja vastupidi.
HTTP	*HyperText Transfer Protocol* – veebi aluseks olev hüpermeediumile suunatud standardne rakenduskihi protokoll.
IP	*Internet Protocol* – esmane võrgustikukihi protokoll.
LAN	*Local Area Network* – arvutivõrk kasutaja territooriumil piiratud geograafilisel alal, enamasti ühes hoones.
NTP	*Network Time Protocol* – protokoll arvutikellade sünkroniseerimiseks.
Puppet DSL	*Puppet Domain-Specific Language* – Puppeti loodud programmeerimis keel.
RSA	*Rivest–Shamir–Adleman* – avaliku võtmega krüptograafiline süsteem.
SDN	*Software Defined Networking* – võrguarhitektuuri kontseptsioon, mis keskendub võrgu konfiguratsiooni programmeerimisele.
SNMP	*Simple Network Management Protocol* – võrgustatud seadmete seire ja halduse protokollistik TCP/IP-mudeli rakenduskihis.
SSH	*Secure Shell* – protokollisar, mis võimaldab turvalist krüpteeritud kaugpöördust.
SSL	*Secure Socket Layer* – krüpteerimisprotokoll.
ZTP	*Zero Touch Provisioning* – kommutaatorite automaatse seadistamise funktsioon.

TFTP	*Trivial File Transfer Protocol* – rakenduskihi protokolli lihtne algkuju peamiselt kettata võrgusõlmede alglaadimiseks.
VLAN	*Virtual Local Area Network* – loogiliselt kokku kuuluv võrgustatud tööjaamade, serverite, võrguseadmete jne kogum.
WSL	*Windows Subsystem for Linux* – Windowsi operatsioonisüsteemi alamsüsteem Linux.
YAML	*YAML Ain't Markup Language* – inimloetav andmete jadastuse keel.

Sisukord

Jooniste loetelu

Tabelite loetelu

1 Sissejuhatus

Tänapäeva digitaalses maailmas on kommutaatorid asutuses üheks oluliseks osaks kohtvõrgus. Kommutaatorid võimaldavad seadmetel, näiteks tööjaamad, serverid, printerid jne, ühenduda kohtvõrku ning pääseda ligi kohtvõrgus olevatele teenustele, näiteks failiserver, meiliserver, jagatud kalender jne. Enamasti kasutatakse asutustes ühe tootja poolt toodetud kommutaatoreid. Ühe tootja kasutamine muudab asutuses kommutaatorite seadistamise ja haldamise efektiivsemaks, kuna erinevate tootjate kommutaatorite seadistamine, näiteks käsurida, erineb.

Lõputöös käsitletud asutuse üheks ülesandeks on ajutise kohtvõrgu loomine. Ajutisi kohtvõrke luuakse eesmärgiga need liita asutuses keskse kohtvõrguga. Selleks kasutab asutus mitmeid Cisco kommutaatoreid. Kommutaatoreid seadistatakse manuaalselt, mis võtab aega ning konfiguratsioonis võib esineda eksimusi. Asutuse jaoks on oluline võimalikult efektiivne ajutise kohtvõrgu jaoks kommutaatorite konfigureerimine. Kommutaatorite konfigureerimise aja ja eksimuste vähendamiseks on mõistlik kogu konfigureerimise protsess automatiseerida.

Töö eesmärk on analüüsida erinevaid võimalusi ajutise kohtvõrgu kommutaatorite automaatseks konfigureerimiseks ning saadud tulemus valideerida. Selle jaoks vaadeldakse erinevaid meetodeid, mis toetavad Cisco kommutaatorite seadistamist. Saadud analüüsi tulemuse valideerimiseks koostatakse prototüüpi ning tuuakse välja tulemused.

2 Lahenduse nõuete analüüs

Peatükis vaadeldakse kuidas, ühe asutuse näitel, konfigureeritakse ajutise kohtvõrgu jaoks kommutaatoreid. Lisaks kirjeldatakse millised on nõuded kommutaatorite automaatseks konfigureerimiseks ja millised on kitsendused.

2.1 Kommutaatorite seadistamine

Kohtvõrguks ehk LAN nimetatakse üksteisega ühendatud seadmete kogumit ühes füüsilises asukohas, näiteks – kontor, kool, kodu. Kohtvõrk koosneb kaablitest, pääsupunktidest, kommutaatoritest, ruuteritest ja teistest komponentidest, mis võimaldavad ühenduda internetti. Kohtvõrk võimaldab seadmetel edastada üksteisele faile, printida jagatud printerisse jne [1]. Ajutist kohtvõrku kasutatakse ajutistes asukohas kohtvõrgu loomiseks, näiteks konverentsidel või LAN pidudel.

Ajutise kohtvõrgu loomiseks kasutab asutus Cisco IOS operatsioonisüsteemiga kommutaatoreid. Cisco Systems on üks tuntumaid võrguseadmete ja tarkvara tootjaid. Kommutaator on võrguseade, mis võtab vastu võrgupakette ja saadab neid edasi nende sihtkohta kohtvõrgus. Asutusel on ajutise kohtvõrgu loomiseks kasutusel 20 kuni 30 kommutaatorit. Kasutatavad Cisco kommutaatorid on põhiliselt erineva portide arvuga ning osad kommutaatorid on ka virnastatavad. Virnastatavad kommutaatorid on kaks või mitu kommutaatorit ühendatud omavahel üheks loogiliseks seadmeks. Nii saab kommutaatori portide arvu suurendada [2]. Kommutaatoritesse konfigureeritakse erinevad virtuaalsed kohtvõrgud ehk VLAN-id (*Virtual Local Area Network*). Kõikide kommutaatorite konfigureerimine toimub käsitsi läbi sidepordi, kas siis kopeerides konfiguratsiooni fail seadmesse või kopeerides käsud seadmesse. Manuaalselt kommutaatorite ükshaaval seadistamine võtab aega ja võib tekkida konfiguratsioonis eksimusi. Kommutaatorite konfigureerimise aja ja eksimuste vähendamiseks on mõistlik kogu protsess automatiseerida.

2.2 Nõuded

Väljatöötatav lahendus peab vastama järgmistele nõuetele:

- Peab võimaldama konfigureerida erineva portide arvuga Cisco kommutaatoreid;
- Peab võimaldama konfigureerida virnastatavaid Cisco kommutaatoreid;
- Võimalik seadistada mitut erinevat Cisco kommutaatorit samaaegselt;
- Eelistatud on vaba ja avatud lähtekoodiga lahendused;
- Lahendus peab olema mastabeeritav;

Portide arvust oleneb millised pordid tuleb konfigureerida juurdepääsu pordiks ja milline üleslüliks.

Virnastatavatel kommutaatoritel on erinev võrguliidese nimede struktuur. Näiteks *GigabitEthernet1/0/1* on virnastatava kommutaatori esimese võrguliidese nimi, *GigabitEthernet0/1* mitte virnastatavate kommutaatoritel.

Mitme kommutaatori samaaegne seadistamine võimaldab aega kokku hoida.

3 Võrguseadmete automaatse konfigureerimise võimalused

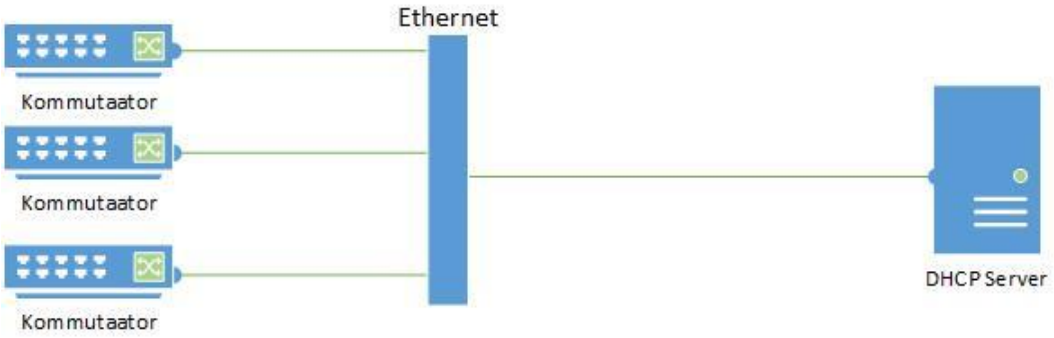
Võrgu automatiseerimine on metoodika milles tarkvara automaatselt konfigureerib, sätestab, haldab ja testib võrguseadmeid. Seda kasutatakse tõhususe tõstmiseks, inimlikke vigade ja tegevuskulude vähendamiseks [3]. Antud peatükis vaadeldakse erinevaid viise, kuidas on võimalik Cisco kommutaatoreid automaatselt konfigureerida. Peatüki lõpus analüüsitakse, milline viis sobib väljatöötatava lahenduse realiseerimiseks kõige paremini.

3.1 Nullpuute Varustamine

Nullpuute varustamine ehk ZTP (*Zero Touch Provisioning*) automatiseerib tarkvara paigaldamist või uuendamist ja konfiguratsiooni paigaldamise protsessi võrguseadmetes, mis võetakse esmakordselt kasutusele kohtvõrgus.

Kui Cisco võrguseade, mis toetab ZTP funktsionaalsust, käivitub ja ei leia konfiguratsiooni faili, lülitub võrguseade ZTP režiimi. ZTP režiimis hakkab seade otsima DHCP (*Dynamic Host Configuration Protocol*) serverit. Serveri leidmisel, küsib seade omale IP-aadressi, lüüsi aadressi, DNS (*Domain Name System*) serveri IP (*Internet Protocol*) ja lubatakse *Guestshell* [4]. *Guestshell* on virtualiseeritud Linuxi põhine keskkond, mis on loodud Linuxi rakenduste käivitamiseks ning sisaldab Python-it seadmete haldamise automatiseerimiseks. [5]. Lisaks IP-aadressile, lüüsi aadressile ja DNS serveri aadressile, on DHCP serveris defineeritud ka HTTP (*HyperText Transfer Protocol*) või TFTP (*Trivial File Transfer Protocol*) aadress, kus asub Pythoni skript, millega konfigureeritakse võrguseadet. Võrguseade laeb DHCP serverilt saadud HTTP või TFTP serveri aadressilt alla Pythoni skripti ning käivitab selle *Guestshell*-is [4]. Pythoni skript sisaldab võrguseadme operatsioonisüsteemi käsurea käske. Kasutades ZTP funktsiooni saab automaatselt seadistada mitmeid kommutaatoreid korraga.

Joonisel 1 on välja toodud nullpuute varustamine ehk ZTP.

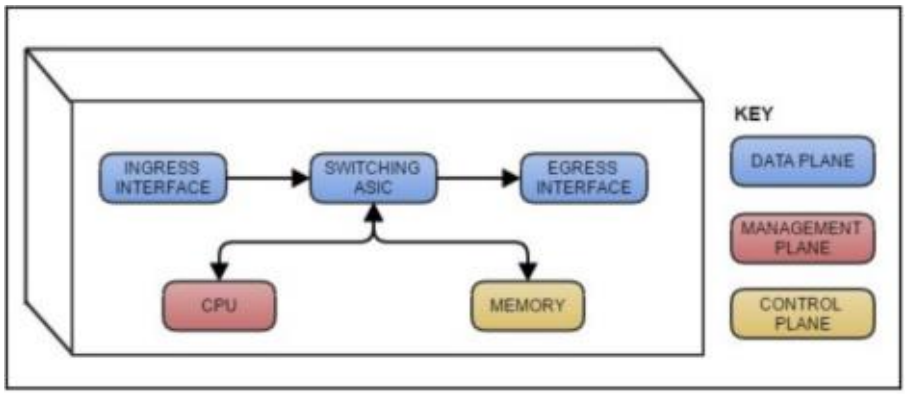


Joonis 1. Nullpuute varustamine

3.2 Programmeeritav võrgustus

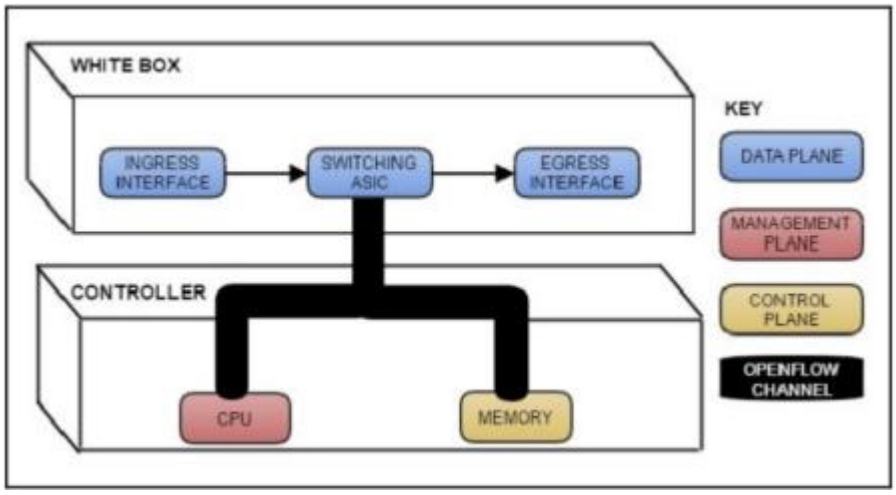
Programmeeritav võrgustus ehk SDN (*Software Defined Networking*) lubab võrgu käitumist programmeerida tsentraalselt kontrollitud viisil tarkvararakenduste kaudu, kasutades rakendusliidest ehk API-t (*Application Programming Interface*). Rakendades SND reguleerimise kihti, saavad võrguadministraatorid hallata tervet võrku ja seadmeid järjekindlalt, sõltumata alusvõrgu tehnoloogia keerukusest [6].

Traditsiooniline võrguarhitektuur koosneb kolmest tasandist – juhtimis-, andme- ja haldustasand (Joonis 2). Juhtimistasand otsustab kuidas võrguliiklust käidelda, andmetasand edastab võrguliikluse vastavalt juhtimistasandi otsusele ja haldustasand on seadme haldamiseks [7].



Joonis 2. Traditsiooniline võrgu arhitektuur [7]

SDN peamine omadus on juhtimistasandi eraldamine andmetasandist (Joonis 3). Juhtimistasand on kommutaatorist lahti ühendatud tsentraliseeritud kontrolleriks, mis on ühenduses kommutaatoriga läbi turvalise kanali [7].



Joonis 3. SDN arhitektuur [7]

Openflow on üks enim levinuid protokoll kontrolleri ja kommutaatori vahel. *Openflow* võimaldab kontrolleril suunata seadme suunamisfunktsioone turvalise kanali kaudu. Kontrollerid pakuvad tsentraliseeritud vaadet ja võimaldavad administraatoril dikteerida ruuteritele ja kommutaatoritele, kuidas võrguliiklust käidelda [8].

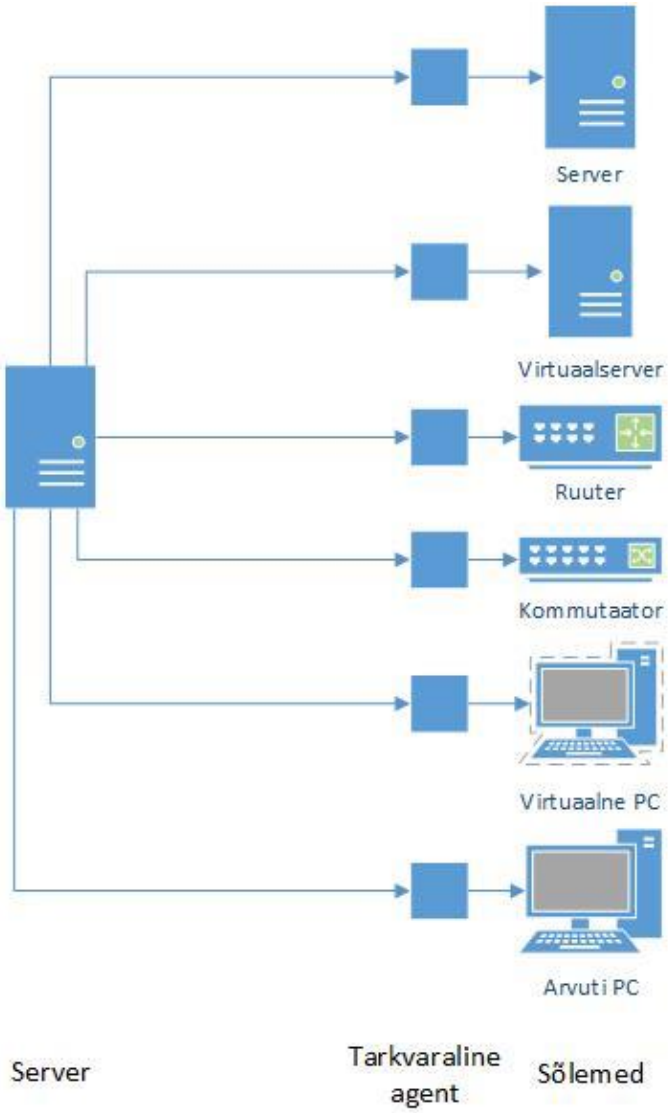
3.3 Konfiguratsioonihalduse instrumendid

Konfiguratsioonihaldus on arvutisüsteemide, serverite ja tarkvara soovitud oleku säilitamise protsess. See protsess hoiab ära, et suured või väiksed muutused jäävad dokumenteerimata. Traditsiooniliselt hallati iga seadme konfiguratsiooni manuaalselt või süsteemiadministraatori poolt koostatud kohandatud skriptidega [9]. Kasutades konfiguratsioonihalduse instrumente määrab süsteemiadministraator soovitud seadme konfiguratsiooni oleku ning instrument viib seadme konfiguratsiooni oleku vastavusse määratud olekuga [10]. Konfiguratsioonihalduse instrumendid muudavad süsteemihalduse ettearvatavaks ja mastabeeritavaks, muudatuste tegemise ja laiali jagamise kiiremaks ja kõrvaldada potentsiaalsed inimlikud eksimused. Lisaks aitavad instrumendid süsteemi auditit läbi viia, küsides seadme oleku kohta informatsiooni [9]. Kuna kõik konfiguratsiooni oleku failid asuvad ühes kohas, saab instrumente hõlpsasti ühildada mõne versioonihaldus süsteemiga [10].

On olemas kahte tüüpi konfiguratsioonihalduse instrumente, agendipõhised ja agendivabad instrumendid. Agendipõhised instrumendid vajavad agendi paigaldamist hallatavasse süsteemi, agendivabad ei nõua hallatavasse süsteemi agendi ega tarkvara

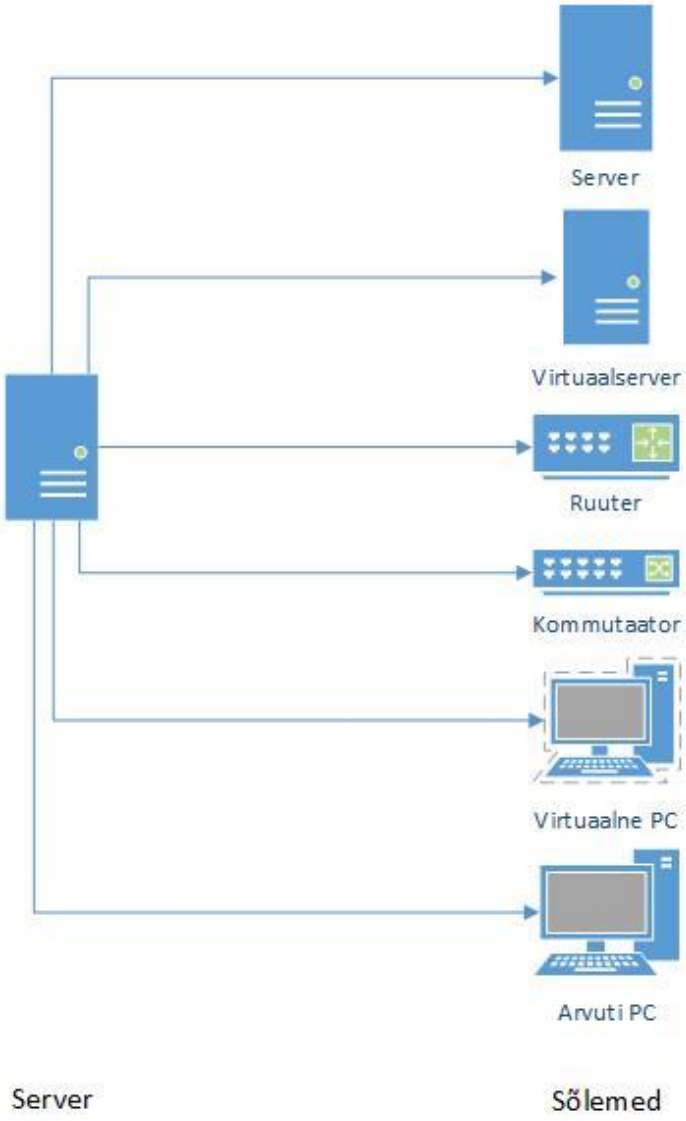
paigaldamist [10]. Agendivabades instrumentides kasutatakse SSH (*Secure Shell*) protokolli hallatava süsteemi ühendamiseks.

Joonisel 4 kujutab agendipõhist seadistust, kus server tähistab füüsilist või virtuaalset serverit kuhu on paigaldatud instrumendi serveri tarkvara. Serveris paikevad seadmete konfiguratsiooni olekute failid. Kastid tähistavad tarkvaralist agenti, mis on paigaldatud instrumendi poolt hallatavatesse seadmetesse. Sõlmed tähistavad seadmeid, mida hallatakse instrumendi serveri poolt.



Joonis 4. Agendipõhised konfiguratsioonihalduse instrumendid

Joonisel 5 kujutab agendivaba seadistust, kus server tähistab füüsilist või virtuaalset serverit kuhu on paigaldatud instrumendi serveri tarkvara. Serveris paikevad seadmete konfiguratsiooni olekute failid. Suhtlus sõlmedega ehk seadmetega, mida instrument haldab, toimub läbi SSH protokolli.



Joonis 5. Agendivabad konfiguratsioonihalduse instrumendid

3.4 Võrgu haldus meetodi valimine

Eelnevates peatükkides vaadeldi kolme moodust kommutaatorite automaatseks konfigureerimiseks – ZTP funktsiooni kasutamine kommutaatorites, SDN openflow kontrolleri ühendamine kommutaatoritega ja kommutaatorite haldamine konfiguratsioonihalduse instrumendiga.

Esimesena käsitleti ZTP funktsiooni, mis võimaldab kommutaatori käivitumisel DHCP serverist alla laadida Pythoni skript, millega konfigureeritakse kommutaator. Antud lahenduse puuduseks on ZTP režiimi lülitumine ainult siis, kui kommutaatori käivitamisel konfiguratsiooni fail puudub. Kommutaatori konfiguratsiooni hiljem muutmiseks tuleb kasutada teisi vahendeid või manuaalselt käsurealt seadistada. ZTP

funktsiooni kasutamine ei nõua eelnevalt kommutaatorite seadistamist, mis on antud lahenduse eeliseks. Asutuses ajutise kohtvõrgu loomiseks kasutatavad Cisco kommutaatoritel puudub *guestshell*, seega ei saa ZTP funktsiooni kasutada.

Teisena vaadeldi SDN tehnoloogiat, millega on võimalik võrgu käitumist programmeerida tsentraalselt kontrollerist. Antud lahenduse eeliseks on kõikide kommutaatorite haldamine tsentraalselt ühest kohast. Kuna kõik Cisco kommutaatorid ei toeta *openflow* võimekust, on see lahenduse puuduseks. SDN tehnoloogiat kasutatakse põhiliselt andmekeskustes.

Viimasena vaadeldi konfiguratsioonihalduse instrumente, mis võimaldavad kommutaatori konfiguratsiooni olekut defineerida. Antud lahenduse eeliseks on võimalus kommutaatorilt pärida informatsiooni ning kasutada seda muutujatena, näiteks kommutaatori portide arv.

Erinevate kommutaatorite automaatseks, samaaegseks konfigureerimiseks on oluline võimalus pärida kommutaatoritelt informatsiooni ning seda salvestada muutujatena. Versioonihaldus süsteemiga hõlbus ühildamine annab ülevaate seadmete oleku ajaloost. Lähtudes püstitatud probleemist ja võimalusi arvestades on parim lahendus konfiguratsioonihalduse instrumendid.

4 Lahenduse tarkvara analüüs ja valik

Seoses peatükis 3.4 tehtud valikuga on vajalik valida sobiv konfiguratsioonihalduse instrumendi tarkvara. Peatükis vaadeldakse erinevaid konfiguratsioonihalduse instrumente, mis võimaldavad kommutaatoreid konfigureerida automaatselt. Lähtudes püstitatud lahenduse nõuetest, vaadeldakse nelja populaarsemat avatud lähtekoodiga tarkvara – Ansible, Chef, Puppet ja Saltstaack. Analüüs teostatakse kasutades kirjalike materjale.

4.1 Ansible

Ansible avaldati esmakordselt Michael DeHaani poolt, aastal 2012, kõrval projektina. Nüüdseks on see kasvanud üheks populaarsemaks konfiguratsioonihalduse instrumendiks [11]. Tänapäeval arendab Ansible tarkvara Red Hat.

Ansible on tarkvara, mis automatiseerib varustamist, konfiguratsiooni haldamist, tarkvara juurutamist, orkestreerimist ja palju muud. Vaikimisi kasutab Ansible SSH protokolli väikeste programmide välja saatmiseks, mida nimetatakse Ansible *Playbook*-ideks. Playbook-id koosnevad tegevustest ehk *task*-idest, mida kirjutatakse YAML (*YAML Ain't Markup Language*) keeles ning kirjeldavad soovitud süsteemi olekut. Programmid kustutatakse pärast nende lõpetamist [12].

Ansible agendivaba omadus lubab seadme või serveriga, mis toetab SSH protokolli, ühenduda ilma, et seadmes oleks paigaldatud spetsiaalne tarkvara.

4.2 Chef

Chef koosneb kolmest komponendist, Chef server, tööjaamad ja sõlmed [13].

Chef server on suhtlusrajaks tööjaamade ja sõlmede vahel. Kõik konfiguratsiooni failid, kokaraamatud ehk *cookbook*-id ja metaandmed on koostatud tööjaamades ning hoiustatakse Chef serveris. Kokaraamatud sisaldavad väärtusi ja informatsiooni soovitud sõlme olekuks. Samuti Chef server hoiab kõikide sõlmede oleku kohta teavet. Enne

muudatuste edastamist autendib Chef server kogu REST API suhtluse, kasutades avaliku võtme krüptimist [13].

Tööjaamad on personaalsed või virtuaalsed arvutid, kus kogu konfiguratsioonikoodi luuakse, testitakse ja muudetakse. Tööjaamu võib olla mitu [13].

Sõlmed on seadmed mida hallatakse Chef-i poolt. Chef-iga saab hallata sõlmesid, mis on virtuaalsed serverid, konteinerid, võrguseadmed ja mäluseadmed. Nendesse seadmetesse tõugatakse konfiguratsiooni muudatused. Igas sõlmes peab olema Chef klient paigaldatud, et viia läbi sammud, mis on vajalikud sõlme viimiseks *cookbook*-i poolt määratletud olekusse [13].

Chef *cookbook*-i alusel hallatakse iga sõlme konfiguratsiooni. Chef server ja Chef klient kindlustavad, et kirjeldatud olek saavutatakse. *Cookbook*-id on kirjutatud Ruby keeles [13].

4.3 Puppet

Puppet on väljatöötatud Puppet Labs-i poolt, mille asutas Luke Kanies aastal 2005. Puppet on kirjutatud Ruby keeles ning on avatud lähtekoodiga tarkvara [14]. Puppet on konfiguratsioonihalduse instrument, millega on võimalik määrata süsteemi olekut.

Puppet kasutab ülem-alluv-mudelit, kus ülem ja alluv suhtlevad läbi turvalise SSL-iga (*Secure Socket Layer*) krüpteeritud kanali. Puppet serveris asuvad failid, mida nimetatakse *manifest*-ideks. *Manifest*-id on kirjutatud Puppet DSL (*Puppet Domain-Specific Language*) keeles ning kirjeldavad soovitud süsteemi olekut. Sõlmedes peab olema paigaldatud Puppet agent, mis suhtleb Puppet serveriga. Iga teatud aja tagant, vaikimisi 30 minutit, küsib sõlmes seadistatud Puppet agent serverilt konfiguratsiooni. Konfiguratsioonis muudatuste korral viib agent muudatused täide ja teavitab serverit [15].

4.4 Saltstack

Saltstack ehk Salt on Thomas Hatchi poolt, aastal 2011, algatatud avatud lähtekoodiga projekt. Algselt oli Salt ette nähtud välkkiireks kaugjuhtimissüsteemiks [16]. Salt-ist on nüüdseks arenenud kaughaldusraamistikuks ja konfiguratsioonihalduse süsteemiks. Salt ja tema kaughaldus raamistik on kirjutatud Pythoni keeles. Salt server saadab välja faile,

mis kirjeldavad kuidas süsteem peab olema konfigureeritud, neid kirjeldusi nimetatakse olekuteks (*states*) ning hoitakse YAML failides nimega SLS (*salt states*) [17].

Salt koosneb serverist ehk *master*-ist ja ühest või mitmest klientidest ehk *minion*-ist. Käske saadetakse serverist klientide sihtrühmale, mis siis täidetakse ja saadetakse andmed tagasi serverise [18].

4.5 Haldustarkvara võrdlus

Tabelis 1 on välja toodud nelja vaadeldava konfiguratsioonihalduse instrumendi omadused.

Tabel 1. Konfiguratsioonihalduse instrumentide võrdlus

	Ansible	**Chef**	**Puppet**	**Saltstack**
Avaldamise aasta	2012	2009	2005	2011
Master/Agent	Jah/Ei	Jah/Jah	Jah/Jah	Jah/Jah
Konfigureerimis keel	YAML	Ruby DSL	Puppet DSL	YML
Mastabeeritav	Jah	Jah	Jah	Jah
Programmeerimiseoskuse vajalikkus haldamisel	Ei	Jah	Jah	Ei
Cisco IOS tugi	Jah	Ei	Jah, nõuab lisa mooduli paigaldamist Puppet serverisse	Jah, nõuab Salt Proxy kasutamist
Cisco IOS moodulite arv	23	-	30	7

4.6 Haldustarkvara valik

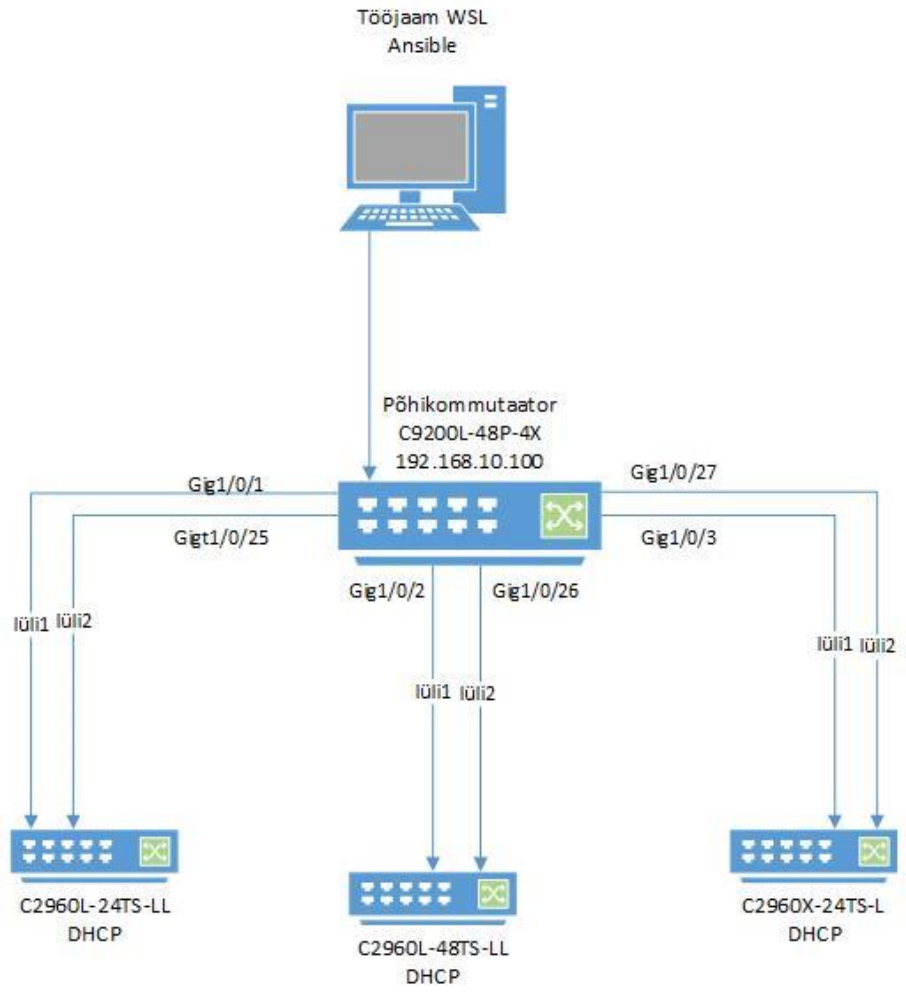
Kõik Cisco kommutaatorid ei toeta agent tarkvara paigaldamist seega on oluline, et konfiguratsioonihalduse instrument suhtleks kommutaatoriga kasutades SSH protokolli. Kuna eelnevas punktis ei tõusnud esile sobilik tarkvara, vaadeldakse tarkvarade populaarsust. Populaarsuse järgi on võimalik näha tarkvara kogukonda. Google Trends abil vaadeldakse kõigi nelja tarkvara populaarsust, millest selgub, et Ansible on kõige populaarsem konfiguratsioonihalduse instrument (vt Lisa 1).

Asutuses kasutatakse Ansible-t virtuaalserverite haldamiseks. Võttes arvesse asutuses töötava personali kogemusi Ansible tarkvara kasutamises ning tarkvara üldist populaarsust on Ansible eelistatud valik.

5 Prototüübi tegemine ja valideerimine

Peatükk käsitleb väljatöötatud prototüübi kirjeldust ning eelseadistamist. Prototüübi valideerimiseks kasutatakse ühte põhikommutaatorit ja kolme automaatselt konfigureeritavat kommutaatorit. Sõltuvalt põhikommutaatori portide arvust, võimaldab prototüüp automaatselt konfigureerida maksimaalselt põhikommutaatori portide arv jagatud kahega, ehk prototüübis kasutatava põhikommutaatori näitel 24 kommutaatorit. Põhikommutaatori portide arvu jagamine kahega tuleneb kahe kaabli kasutamisest iga konfigureeritava kommutaatori kohta. Kahte kaablit kasutatakse ainult konfigureerimise ajal, pärast konfigureerimise lõpetamist kasutatakse ühte kaablit, mis on üleslingiks.

Joonisel 6 on välja toodud prototüübi rakendamisel kasutatav topoloogia.



Joonis 6. Prototüübi topoloogia

5.1 Prototüübis kasutatavad füüsilised seadmed

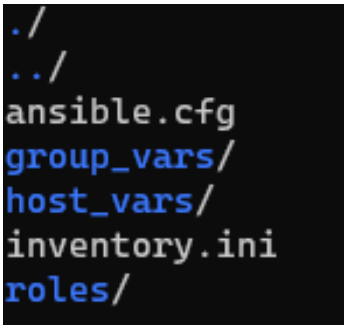
Prototüübi tegemisel kasutatakse 4 kommutaatorit, millest 1 on põhikommutaator ja 3 konfigureeritavad kommutaatorid. Põhikommutaatoriks kasutatakse Cisco Catalyst 9200L-48P-4X seadet. Põhikommutaatori porti 1–3(Joonis 6 *Gig1/0/1–Gig1/0/3*) on seadistatud VLAN10. Põhikommutaatorisse on seadistatud DHCP server, mis jagab VLAN10-sse IP-aadresse. DHCP vahemikuks kasutatakse 192.168.10.200–192.168.10.254. Portidesse 1–3 on ühendatud 3 kommutaatorit, mida Ansible-ga konfigureeritakse (Joonis 6 lüli1). Prototüübis kasutatakse erineva portide arvuga vaikekonfiguratsioonis Cisco kommutaatoreid: C2960L-24TS-LL, C2960L-48TS-LL ja C2960X-24TS-L. Kaks kasutatavatest kommutaatoritest on virnastatavad.

Prototüübis on võetud konfigureeritavate kommutaatorite üleslingiks viimane port, mis on ühendatud põhikommutaatorisse (vt Joonis 6 lüli2). Kommutaatorite staatilise IP seadistamiseks võetakse kasutusele VLAN20, kuna Cisco kommutaatorites mitmele võrguliidesele ei ole võimalik seadistada IP-aadressi samast alamvõrgust.

5.2 Ansible tarkvara paigaldamine ja seadistamine

Tarkvara lahenduse rakendamisel kasutatakse operatsioonisüsteemi Windows alamsüsteemi Linux-i ehk WSL (*Windows Subsystem for Linux*) distributsiooni Ubuntu. Alamsüsteemis on paigaldatud Ansible versioon 2.9.5 ja Python versioon 3.6.9 pakid. Ansible kaust seadistatakse kasutades Ansible parimaid praktikaid.

Joonis 7 kujutab Ansible kausta sisu lähtudes parimatest praktikatest.



Joonis 7. Ansible kaust

- *Ansible.cfg* – Sisaldab Ansible rakenduse seadeid;
- *group_vars* – Kaust sisaldab grupi põhised muutujaid;
- *host_vars* – Kaust sisaldab seadme põhiseid muutujaid;

- *inventory.ini* – Sisaldab Ansible poolt hallatavate seadmete nimekirja;
- *roles* – Kaust sisaldab tegevuste kogumit;

Joonis 8 kujutab Ansible kaustas *ansible.*cfg faili sisu. Prototüübis kasutatakse Ansible funktsiooni *set_fact*, mis määrab *playbook*-i käigus muutuja ning salvestab ajutiselt mällu. Määratud *fact*-i püsivalt kasutamiseks tuleb *ansible.cfg* failis määrata *fact*-i salvestamise formaat ning asukoht. *Fact_caching* määrab ära faktide salvestamise formaadi ning *fact_caching_connection* määrab kuhu *fact*-id salvestatakse. Lisaks määratakse *stdout_callback* sättele väärtus yaml, mis muudab Ansible väljundi rohkem arusaadavamaks.

```
[defaults]
host_key_checking = false
stdout_callback = yaml
fact_caching = yaml
fact_caching_connection = /tmp/facts_cache
```

Joonis 8. Faili ansible.cfg sisu

Inventory.ini fail nimetatakse Ansible-s ümber *task01_inv.ini*-ks, kuna prototüüp koosneb mitmest etapist, mis kasutavad erinevat seadmete nimekirja faili. Konfiguratsioonis määratakse *task01_inv.ini* failis põhikommutaatori IP-aadressiks 192.168.10.100.

Joonis 9 kujutab *group_vars* kaustas *all.yml* faili sisu. *Group_vars* kataloogi luuakse fail *all.yml,* kus määratakse kasutajakonto ja seadmesse ühendamisega seotud muutujad, mida kasutatakse kõikide Ansible poolt hallatavates seadmetes. Ansible nõuab kommutaatori operatsioonisüsteemi määramist muutujates ennem ühendamist kommutaatorisse. Asutuses kasutatakse ajutise kohtvõrkude loomiseks enamasti Cisco IOS (*Cisco Internetwork Operating System*) operatsioonisüsteemiga kommutaatoreid, seega määratakse võrguseadme operatsioonisüsteemi muutujaks IOS.

```
ansible_connection: network_cli
ansible_network_os: ios
ansible_user: admin
ansible_password: Parool.123
```

Joonis 9. Faili all.yml sisu

Joonis 10 kujutab *group_vars* kaustas *switches.yml* faili sisu. *Group_vars* kaustas faili *switches.yml* määratakse kõik kommutaatorite ühised muutujad. Ühisteks muutujateks on haldus liidese VLAN, lüüsi aadress ja võrgumask, SNMP (*Simple Network*

Management Protocol) serveri aadress ja kooskonnastring ning NTP (*Network Time Protocol*) serveri aadress.

```
mgmt_vlan: 20
mgmt_gw: 192.168.20.1
mgmt_netmask: 255.255.255.0

snmp_community: test.community
snmp_server: 192.168.0.20

ntp_server: 192.168.0.21
```

Joonis 10. Faili switches.yml sisu

Joonis 11 kujutab *group_vars* kaustas *tallinn.yml* faili asukoha Tallinn muutujad. Kommutaatoritele asukoha põhiste erisuste määramiseks kasutatakse *group_vars* kaustas asukoha nimelisi muutujate faile. Asukoht lisatakse iga *playbook*-i käivitamisel lisa muutujana.

```
vlans:
  - name: test_vlan1
    id: 11
  - name: test_vlan2
    id: 12
  - name: tallinn
    id: 15

access_vlan:
  name: tallinn
  id: 15
```

Joonis 11. Faili tallinn.yml sisu

5.3 Kommutaatorite SSH ligipääsu seadistamine

Ansible kasutab hallatavate kommutaatoritega suhtlemiseks SSH protokolli, kuid vaikekonfiguratsioonis kommutaatoritel ei ole lubatud üle SSH ühendamine. Selleks, et Ansible saaks kommutaatoritega suhelda on vaja kommutaatorites seadistada kasutajakonto ning luua RSA (*Rivest–Shamir–Adleman*) võti. SSH ligipääsu seadistamiseks kasutatakse ZTP sarnast funktsiooni Cisco *autoinstall*. *Autoinstall* kasutab DHCP serveris määratud skripti, mis asub TFTP serveris. TFTP serveriks kasutatakse põhikommutaatorit, kus paikneb konfiguratsiooni skript (Joonis 12) ning DHCP sätetes on määratud skripti asukoht. Vaikekonfiguratsioonis kommutaatori käivitumisel laetakse

alla TFTP serverist skripti ning seadistatakse SSH ligipääs. Edasine tegevus toimub tööjaamas Ansible-ga.

```
ip domain-name example.com
crypto key generate rsa modulus 2048
ip ssh version 2
service password-encryption
username admin privilege 15 password Parool.123
line vty 0 15
login local
logging synchronous
transport input ssh
end
```

Joonis 12. SSH ligipääsu seadistamise skript

5.4 Ansible esimene *playbook*

Esmalt selgitatakse välja kommutaatoritele antud IP-aadressid. DHCP poolt antud IP-aadresside teada saamiseks küsitakse Ansible-ga põhikommutaatorilt ARP väljundit ning salvestatakse väljund faili. ARP väljundis kuvatakse kõik VLAN10-s olevad IP-aadressid, kaasa arvatud ka põhikommutaatori ja lüüsi aadress. Faili sisust filtreeritakse välja ainult automaatselt konfigureeritavate kommutaatorite IP-aadressid, kasutades regulaaravaldist (Joonis 13). Kuna DHCP vahemik on seadistatud alates 192.168.10.200 aadressist, võib kindel olla, et kõik välja filtreeritud aadressid on konfigureeritavate kommutaatorite omad. Saadud kommutaatorite IP-aadresside põhjal genereeritakse uus seadmete nimekirja fail nimega *task02_inv.ini*. Kõik leitud aadressid lisatakse seadmete nimekirja failis gruppi *switches*, et kõikidele seadmetele saaks ühiseid muutujaid määrata.

```
- name: Remove gateway and core-sw ip from list
  shell: |
      cat /home/mikk/arp.txt | grep -oE " 192.168.10.[2][0-9][0-9]" >
      /home/mikk/addresses.txt
  delegate_to: localhost
```

Joonis 13. ARP väljundi filtreerimise regulaaravaldis

5.5 Ansible teine *playbook*

Teine Ansible *playbook* koosneb neljast tegevusest. Järgnevalt kirjeldatakse neid nelja tegevust lähemalt.

5.5.1 Võrguliidese nime filtreerimine ning üleslingi ja IP seadistamine

Joonis 14 kujutab kommutaatori portide arvu leidmise loogikat. Esimese tegevusena selgitatakse välja iga kommutaatori portide arv ning võrguliidese nimeosa ja numbriosa. Portide arvu saamiseks kasutatakse Ansible *gather_facts* funktsiooni, mis kogub iga kommutaatori kohta informatsiooni ning salvestab muutujateks. Kogutud faktidest loetakse kokku kõigi võrguliideste nimed. Saadud numbrist tuletatakse kommutaatori portide arv. Kui saadud number ületab arvu 48 määratakse kommutaatori portide arvuks 48, kui number on väiksem, määratakse portide arvuks 24. Määratud portide arvud salvestatakse muutujateks.

```
- set_fact:
    last_port:  48
    cacheable: yes
  when: ansible_net_interfaces | select('match', '^(Gig|Eth)') | list
| count > 49

- set_fact:
    last_port:  24
    cacheable: yes
  when: ansible_net_interfaces | select('match', '^(Gig|Eth)') | list
| count < 30
```

Joonis 14. Kommutaatori portide arvu leidmine

Cisco seadmete võrguliidese nimi sõltub kommutaatori mudelist ning kas kommutaator on virnastatav või mitte. Võrguliidese nimi koosneb kahes osast, nimeosa ja numbriosa. Nimeosa tuleneb üldjuhul võrguliidese maksimum kiirusest näiteks *Ethernet* – 100 *megabit per second, GigabitEthernet* – 1 *gigabit per second, TenGigabitEthernet* – 10 *gigabit per second*. Numbriosa sõltub, kas kommutaator on virnastatav või mitte, näiteks Cisco virnastatavate kommutaatorite esimese pordi võrguliidese nime numbriosa on 1/0/1. Esimene number tähistab kommutaatori järjekorra numbrit virnas. Teine number tähistab mooduli numbrit, ning kolmas tähistab pordi numbrit. Mitte virnastatavate kommutaatoritel esimene number puudub. Iga kommutaatori ühe võrguliidese nime väljaselgitamine on oluline, kuna võrguliidese nimi võib varieeruda, ning automaatseks konfigureerimiseks on oluline teada iga kommutaatori võrguliidese nimekuju.

Joonis 15 kujutab võrguliidese nime numbriosa ja nimeosa leidmise loogikat. Kommutaatorite võrguliideste nimekuju väljaselgitamiseks võetakse eelnevalt kogutud faktidest kõigi võrguliideste nimede list ning valitakse üks. Pythoni funktsiooni *split* abil

jagatakse saadud nimi, sõna „Ethernet“ põhjal kaheks osaks. Esimeseks osaks saadakse poolik võrguliidese nimeosa ning teiseks osaks numbriosa. Täieliku nimeosa saamiseks jagatakse võrguliidese nimi numbriosa põhjal kaheks ning valitakse esimene. Saadud täielik nimeosa ja numbriosa salvestatakse muutujateks.

```
- set_fact:
    interface_name_split: 'Ethernet'
    interface_list: "{{ ansible_net_interfaces | select('match',
'^(Gig|Eth)') | list | last }}"
    cacheable: yes

- set_fact:
    interface_name_nr: '{{
interface_list.split(interface_name_split)[1] }}'
    cacheable: yes

- set_fact:
    interface_name: "{{ interface_list.split(interface_name_nr)[0] }}"
    cacheable: yes
```

Joonis 15. Võrguliidese nimeosa ja numbriosa leidmine

Joonis 16 kujutab võrguliides nime numbriosa jagamist. Kommutaatorite võrguliideste seadistamiseks on vaja võrguliideste nime numbriosa jagad osadeks. Võrguliidese nime numbriosa jagatakse sümboli „/“ põhjal kaheks või kolmeks, sõltuvalt kas kommutaator on virnastatav või mitte, ning loetakse osad kokku. Saadud tulemus salvestatakse muutujaks. Võrguliidese nime numbriosa arvu kasutatakse numbriosa pikkuse välja selgitamiseks.

```
- set_fact:
    split_nr: '/'
    cacheable: yes

- set_fact:
    nr_count: "{{interface_name_nr.split(split_nr) | count }}"
    cacheable: yes
```

Joonis 16. Võrguliidese numbriosa jagamine

Joonis 17 kujutab kommutaatoritele üleslingi seadistamist. Teise tegevusena seadistatakse kommutaatoritele hostinimi, üleslink ja staatilise IP-aadress. Staatilise IP-aadressi määramiseks on vaja seadistada ka kommutaatorile üleslink. Üleslingi seadistamiseks kasutatakse esimeses tegevuses saadud võrguliideste nime osasid. Võrguliidese nime numbriosa pikkus sõltub, kas kommutaator on virnastatav või mitte,

seega esimeses tegevuses kokku leotud numbriosa arvu kasutatakse üleslingi täpse võrguliidese nime saamiseks.

```
- name: Set uplink
  ios_config:
    lines:
     - switchport mode trunk
     - switchport trunk allowed vlan "{{ mgmt_vlan }}"
    parents: interface {{interface_name}}{{
interface_name_nr.split(split_nr) [0] }}/{{
interface_name_nr.split(split_nr) [1] }}/{{last_port}}
  when: nr_count == "3"

- name: Set uplink
  ios_config:
    lines:
     - switchport mode trunk
     - switchport trunk allowed vlan "{{ mgmt_vlan }}"
    parents: interface {{interface_name}}{{
interface_name_nr.split(split_nr) [0] }}/{{last_port}}
  when: nr_count == "2
```

Joonis 17. Üleslingi seadistamine

5.5.2 Seadmete nimekirja faili genereerimine

Kolmanda tegevusena genereeritakse kolmas seadmete nimekirja faili nimega *task03_inv.ini*. Seadmete nimekirja faili genereeritakse automaatselt iga kommutaatori portide arvu, operatsioonisüsteemi, hostinime ja IP-aadressi muutuja. Muutujad saadakse iga kommutaatori faktidest.

5.5.3 Kommutaatorite vaikelüüsi seadistamine

Neljanda tegevusena seadistatakse kommutaatoritele vaikelüüsi aadress. Kuna peale vaikelüüsi seadistamist kaotab Ansible kommutaatoritega ühenduse, tehakse neljas tegevus *playbook*-is viimasena. Kolmandas *playbook*-is kasutatakse kommutaatoritesse ühendamises peatükis 5.5.2 konfigureeritud staatilist IP-aadressi.

5.6 Ansible kolmas *playbook*

Playbook-is seadistatakse kommutaatoritele VLAN-id ja lisatakse need üleslinki ja juurdepääsu portidesse. VLAN-id võetakse *group_vars* kaustast, iga asukoha jaoks eraldi failist. Asukoht määratakse *playbook*-i käivitamisel lisa muutujana. Üleslingi ja

juurdepääsu portide seadistamisel kasutatakse teises *playbook*-is muutujateks määratud võrguliidese nimede osasid. Lisaks seadistatakse *group_vars* kaustas olevas *switches.yml* failis määratud SNMP serveri aadress ja kooskonnastring ning NTP serveri aadress.

Playbook-i lõpus salvestatakse konfiguratsioon ning väljastatakse ekraanil iga kommutaatori hostinimi, asukoht, seerianumber ning üleslingi port (Joonis 18). Väljastatud seeria numbri järgi tuvastatakse füüsiline kommutaator.

```
Switch name=tallinn-sw01 location=tallinn serial_nr=XXXXXX Uplink=24
Switch name=tallinn-sw02 location=tallinn serial_nr=XXXXXX Uplink=48
Switch name=tallinn-sw03 location=tallinn serial_nr=XXXXXX Uplink=24
```

Joonis 18. Konfigureeritud kommutaatorite väljund

5.7 Prototüübi valideerimine

Väljatöötatud Ansible *playbook*-id valideeritakse kolme vaikekonfiguratsioonis kommutaatorit kasutades. Järgnevalt vaadeldakse, kuidas *playbook*-id käivitatakse ning mis tulemuseni jõuti.

Enne Ansible *playbook*-ide käivitamist seadistati vajaminevad muutujad kausta *group_vars* failides. Seejärel käivitati Ansible esimene *playbook*. Joonis 19 kujutab esimese *playbook*-i käivitamise käsku.

```
ansible-playbook -i task01_inv.ini task01_locate_switches.yml -e
location=tallinn
```

Joonis 19. Ansible esimene käsk

Esimese *playbook*-i tulemusena leiti kõik kommutaatorid, mis on põhikommutaatori külge ühendatud ning millega valideeritakse prototüübi tulemust. Järgmiseks seadistati *task02_inv.ini* failis iga leitud kommutaatorile haldus IP-aadress ja hostinimi. Järgmiseks käivitati Ansible teine *playbook*. Joonis 20 kujutab teise playbook-i käivitamise käsku.

```
ansible-playbook -i task02_inv.ini task02_set_mgmt_ip.yml -e
location=tallinn
```

Joonis 20. Ansible teine käsk

Teise *playbook*-i tulemusena seadistati igale kommutaatorile staatiline IP-aadress ja üleslingi port ning vaikelüüsi aadress. *Playbook*-i tulemusena vahetus kommutaatorite IP-aadress. Viimaseks käivitati Ansible kolmas *playbook*. Joonis 21 kujutab kolmanda *playbook*-i käivitamise käsku.

```
ansible-playbook -i task03_inv.ini task03_conf_sw.yml -e
location=tallinn
```

Joonis 21. Ansible kolmas käsk

Kõigi *Playbook*-i käivitamise väljund on välja toodud Lisas 2-4.

Kõikide *playbook*-ide ja *autoinstall* skripti kasutamise tulemusena olid kõik kommutaatorid edukalt konfigureeritud ning olid valmis ajutises kohtvõrgus kasutamiseks. Automaatselt konfigureeritud ühe kommutaatori konfiguratsiooni väljavõte on näha Lisas 5.

Prototüüp täidab kõik nõuded, mis on esitatud esimeses peatükis väljatöötatavale lahendusele. Prototüübis kasutatud Ansible *playbook*-ide kood on kättesaadav https://github.com/mikkaruoja/Ansible lingilt.

6 Kokkuvõte

Käesoleva bakalaureusetöö eesmärk oli välja selgitada kõige sobilikum viis, ühe asutuse näitel, ajutises kohtvõrgus kasutatavate kommutaatorite konfigureerimise automatiseerimiseks ning saadud tulemus valideerida. Selleks analüüsiti kolme võimalust - nullpuute varustamine ehk ZTP, programmeeritav võrgustus ehk SDN ning konfiguratsioonihalduse instrumendid. Analüüsi tulemusel osutus sobilikuks viisiks konfiguratsioonihalduse instrumendid. Valitud viisi rakendamiseks tuli välja selgitada sobilik konfiguratsioonihalduse instrumendi tarkvara. Analüüsiti nelja populaarsemat – Ansible, Chef, Puppet ning Saltstack. Analüüsi tulemusel valiti Ansible. Valitud tarkvara valideerimiseks tehti prototüüp.

Prototüübi tegemisel selgus, et ainult Ansible kasutamine ei võimalda vaikekonfiguratsioonis kommutaatorite automaatset konfigureerimist. Ilmnenud probleem lahendati kasutades ZTP sarnast funktsiooni Cisco *autoinstall*, mis võimaldab skriptiga vaikekonfiguratsioonis kommutaatorit seadistada.

Töö tulemusena töötas autor välja Ansible *playbook*-id ning Cisco *autoinstall* skripti. Kasutades välja töötatud Ansible *playbook*-ide ja Cisco *autoinstall* skripti kombinatsiooni konfigureeriti edukalt kolm vaikekonfigratsioonis kommutaatorit ning olid valmis asutuse ajutises kohtvõrgus kasutamiseks.

Väljatöötatud Ansible *playbook*-e on võimalik kasutada ka teiste Cisco kommutaatorite seadistamiseks. Muutes Ansible *playbook*-i *task*-ides kasutatavaid mooduleid ja operatsioonisüsteemi muutujat näiteks IOSXR-ks, on võimalik seadistada ka Cisco IOS-XR kommutaatoreid.

Autori poolt väljatöötatud Ansible *playbook*-id ja Cisco *autoinstall* skript täidavad kõik püstitatud nõuded ning muudavad asutusel ajutise kohtvõrgu loomisel kommutaatorite konfigureerimise efektiivsemaks ning aitavad vähendada seadistamisel esinevaid vigu.

Kasutatud kirjandus

[1] Cisco, „What Is a LAN?,“ Cisco Systems, [Võrgumaterjal]. Available: https://www.cisco.com/c/en/us/products/switches/what-is-a-lan-local-area-network.html. [Kasutatud 17 05 2020].

[2] Cisco, „Cisco stacking,“ Cisco Systems, 13 02 2020. [Võrgumaterjal]. Available: https://www.cisco.com/c/en/us/support/docs/smb/switches/cisco-350x-series-stackable-managed-switches/smb5252-what-is-stacking.html. [Kasutatud 17 05 2020].

[3] M. Rouse, „Network automation,“ Searchnetworking, October 2017. [Võrgumaterjal]. Available: https://searchnetworking.techtarget.com/definition/network-automation. [Kasutatud 17 05 2020].

[4] Cisco, „Cisco ZTP,“ Cisco Systems, [Võrgumaterjal]. Available: https://developer.cisco.com/docs/ios-xe/#!zero-touch-provisioning/zero-touch-provisioning. [Kasutatud 17 05 2020].

[5] Cisco, „Cisco Guestshell,“ Cisco Systems, 1 04 2019. [Võrgumaterjal]. Available: https://www.cisco.com/c/en/us/td/docs/ios-xml/ios/prog/configuration/166/b_166_programmability_cg/guest_shell.html. [Kasutatud 17 05 2020].

[6] Blueplanet, „What is SDN,“ Blueplanet, [Võrgumaterjal]. Available: https://www.blueplanet.com/resources/What-is-SDN.html. [Kasutatud 17 05 2020].

[7] S. Azodolmolky ja O. Coker, Software-Defined Networking with OpenFlow: Deliver innovative business solutions, Birmingham: Packt Publishing Ltd., 2017.

[8] Cisco, „Cisco openflow,“ Cisco Systems, 05 04 2020. [Võrgumaterjal]. Available: https://www.cisco.com/c/en/us/td/docs/ios-xml/ios/prog/configuration/1611/b_1611_programmability_cg/OpenFlow.html. [Kasutatud 17 05 2020].

[9] Redhat, „What is configuration management?,“ Redhat, [Võrgumaterjal]. Available: https://www.redhat.com/en/topics/automation/what-is-configuration-management. [Kasutatud 17 05 2020].

[10] Networklessons, „Configuration Management Tools and Version Control Systems,“ Networklessons, [Võrgumaterjal]. Available: https://networklessons.com/cisco/evolving-technologies/configuration-management-tools-and-version-control-systems. [Kasutatud 017 05 2020].

[11] M. Heap, Ansible: From Beginner to Pro, New York: Apress, 2016.

[12] Ansible, „How ansible works,“ Ansible, [Võrgumaterjal]. Available: https://www.ansible.com/overview/how-ansible-works. [Kasutatud 17 05 2020].

[13] E. Krout, „A Beginner's Guide to Chef,“ Linode, 18 12 2019. [Võrgumaterjal]. Available: https://www.linode.com/docs/applications/configuration-management/beginners-guide-chef/. [Kasutatud 17 05 2020].

[14] S. L. Kshirsagar, S. V. Chordiya, V. V. Gulumkar ja A. M. Hattarge, „Easier Automation with Puppet,“ *International Journal for Scientific Research & Development,* 2015.

[15] J. Slagle, Learning Puppet Security, Brimingham: Packt Publishing Ltd., 2015.

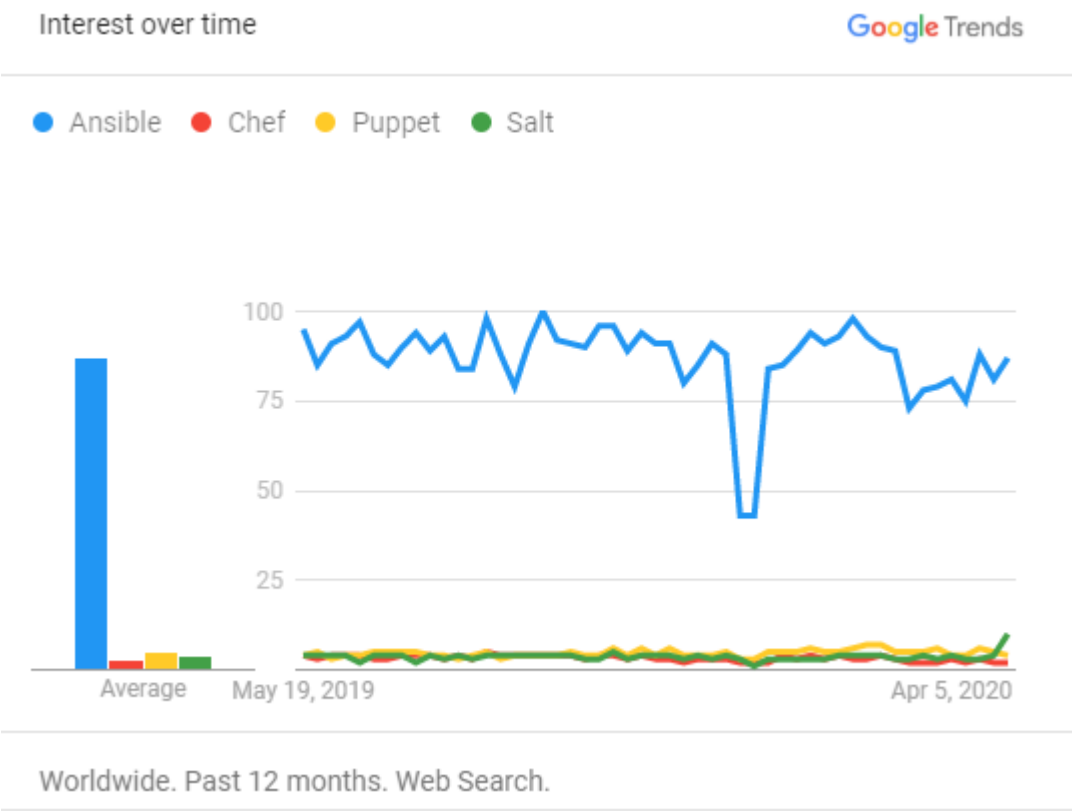
[16] C. Myers, Learning SaltStack, Birmingham: Packt Poblishing Ltd., 2015.

[17] C. Sebenik ja T. Hatch, Salt Essentials: Getting Started with Automation at Scale, California: O'Reilly Media, Inc, 2015.

[18] J. Hall, Mastering SaltStack, Birmingham: Packt Publishing Ltd., 2016.

Lisa 1 – Google Trends

Väljavõte vaadeldavate konfiguratsioonihalduse instrumentide viimase aasta populaarsusest (14.05.2020).



Lisa 2 – Ansible esimese *playbook*-i käivitamise väljund

```
mikk@DESKTOP:$ ansible-playbook -i task01_inv.ini task01_locate_switches.yml -e location=tallinn
PLAY [core]
*********************************************************************************************

TASK [find_switches : Get IPs from arp]
************************************************************
ok: [192.168.10.100]

TASK [find_switches : copy to file on localhost]
*****************************************************
changed: [192.168.10.100 -> localhost]

TASK [find_switches : Remove gateway and core-sw ip from list]
***************************************
changed: [192.168.10.100 -> localhost]

TASK [find_switches : Get ips as variable]
*********************************************************
ok: [192.168.10.100 -> localhost]

TASK [find_switches : Copy addresses to task2_inv]
*****************************************************
changed: [192.168.10.100 -> localhost]

PLAY RECAP
*********************************************************************************************
192.168.10.100   : ok=5    changed=3    unreachable=0    failed=0    skipped=0    rescued=0
ignored=0
```

Lisa 3 – Ansible teise *playbook*-i käivitamise väljund

```
mikk@DESKTOP:$ ansible-playbook -i task02_inv.ini task02_set_mgmt_ip.yml -e location=tallinn
PLAY [switches]
*********************************************************************************

TASK [Gathering Facts]
*********************************************************************************
ok: [tallinn-sw1]
ok: [tallinn-sw2]
ok: [tallinn-sw3]

TASK [get_interface_name : set_fact]
*****************************************************************
skipping: [tallinn-sw1]
skipping: [tallinn-sw3]
ok: [tallinn-sw2]

TASK [get_interface_name : set_fact]
*****************************************************************
skipping: [tallinn-sw2]
ok: [tallinn-sw1]
ok: [tallinn-sw3]

TASK [get_interface_name : set_fact]
*****************************************************************
ok: [tallinn-sw2]
ok: [tallinn-sw3]
ok: [tallinn-sw1]

TASK [get_interface_name : set_fact]
*****************************************************************
ok: [tallinn-sw1]
ok: [tallinn-sw2]
ok: [tallinn-sw3]

TASK [get_interface_name : set_fact]
*****************************************************************
ok: [tallinn-sw2]
ok: [tallinn-sw1]
ok: [tallinn-sw3]

TASK [get_interface_name : set_fact]
*****************************************************************
ok: [tallinn-sw1]
ok: [tallinn-sw2]
ok: [tallinn-sw3]

TASK [get_interface_name : set_fact]
*****************************************************************
ok: [tallinn-sw2]
ok: [tallinn-sw1]
ok: [tallinn-sw3]

TASK [set_name_and_ip : Set hostname]
*****************************************************************
changed: [tallinn-sw1]
```

```
changed: [tallinn-sw3]
changed: [tallinn-sw2]

TASK [set_name_and_ip : Configure MGMT Vlan]
*****************************************************
changed: [tallinn-sw1]
changed: [tallinn-sw3]
changed: [tallinn-sw2]

TASK [set_name_and_ip : Configure interface vlan IP]
***********************************************
changed: [tallinn-sw1]
changed: [tallinn-sw3]
changed: [tallinn-sw2]

TASK [set_name_and_ip : Set uplink]
*************************************************************
skipping: [tallinn-sw1]
changed: [tallinn-sw3]
changed: [tallinn-sw2]

TASK [set_name_and_ip : Set uplink]
*************************************************************
skipping: [tallinn-sw2]
skipping: [tallinn-sw3]
changed: [tallinn-sw1]

TASK [set_name_and_ip : Gather facts]
*************************************************************
ok: [tallinn-sw1]
ok: [tallinn-sw2]
ok: [tallinn-sw3]

TASK [set_name_and_ip : Create host vars file]
*****************************************************
changed: [tallinn-sw1]
changed: [tallinn-sw3]
changed: [tallinn-sw2]

PLAY [localhost]
*********************************************************************************

TASK [populate_inv : Populate inventory]
***********************************************************
changed: [localhost]

PLAY [switches]
*********************************************************************************

TASK [Configure interface vlan IP]
*************************************************************


PLAY RECAP
*********************************************************************************************
localhost     : ok=1    changed=1    unreachable=0    failed=0    skipped=0    rescued=0
ignored=0
tallinn-sw1   : ok=13   changed=5    unreachable=0    failed=1    skipped=2    rescued=0
ignored=0
tallinn-sw2   : ok=13   changed=5    unreachable=0    failed=1    skipped=2    rescued=0
ignored=0
tallinn-sw3   : ok=13   changed=5    unreachable=0    failed=1    skipped=2    rescued=0
ignored=0
```

Lisa 4 – Ansible kolmanda *playbook*-i käivitamise väljund

```
mikk@DESKTOP:$ ansible-playbook -i task03_inv.ini task03_conf_sw.yml -e location=tallinn
PLAY [tallinn]
*********************************************************************************
TASK [Gathering Facts]
*********************************************************************
ok: [tallinn-sw1]
ok: [tallinn-sw2]
ok: [tallinn-sw3]

TASK [conf_switches : Set vtp mode]
*************************************************************
changed: [tallinn-sw1]
changed: [tallinn-sw3]
changed: [tallinn-sw2]

TASK [conf_switches : Configure all Vlans]
*********************************************************
changed: [tallinn-sw1] => (item={'name': 'test_vlan1', 'id': 11})
changed: [tallinn-sw3] => (item={'name': 'test_vlan1', 'id': 11})
changed: [tallinn-sw2] => (item={'name': 'test_vlan1', 'id': 11})
changed: [tallinn-sw1] => (item={'name': 'test_vlan2', 'id': 12})
changed: [tallinn-sw3] => (item={'name': 'test_vlan2', 'id': 12})
changed: [tallinn-sw2] => (item={'name': 'test_vlan2', 'id': 12})
changed: [tallinn-sw1] => (item={'name': 'tallinn', 'id': 15})
changed: [tallinn-sw3] => (item={'name': 'tallinn', 'id': 15})
changed: [tallinn-sw2] => (item={'name': 'tallinn', 'id': 15})

TASK [conf_switches : Add all vlans to Uplink]
*****************************************************
skipping: [tallinn-sw1] => (item={'name': 'test_vlan1', 'id': 11})
skipping: [tallinn-sw1] => (item={'name': 'test_vlan2', 'id': 12})
skipping: [tallinn-sw1] => (item={'name': 'tallinn', 'id': 15})
changed: [tallinn-sw3] => (item={'name': 'test_vlan1', 'id': 11})
changed: [tallinn-sw2] => (item={'name': 'test_vlan1', 'id': 11})
changed: [tallinn-sw3] => (item={'name': 'test_vlan2', 'id': 12})
changed: [tallinn-sw2] => (item={'name': 'test_vlan2', 'id': 12})
changed: [tallinn-sw3] => (item={'name': 'tallinn', 'id': 15})
changed: [tallinn-sw2] => (item={'name': 'tallinn', 'id': 15})

TASK [conf_switches : Add all vlans to Uplink]
*****************************************************
skipping: [tallinn-sw2] => (item={'name': 'test_vlan1', 'id': 11})
skipping: [tallinn-sw2] => (item={'name': 'test_vlan2', 'id': 12})
skipping: [tallinn-sw2] => (item={'name': 'tallinn', 'id': 15})
skipping: [tallinn-sw3] => (item={'name': 'test_vlan1', 'id': 11})
skipping: [tallinn-sw3] => (item={'name': 'test_vlan2', 'id': 12})
skipping: [tallinn-sw3] => (item={'name': 'tallinn', 'id': 15})
changed: [tallinn-sw1] => (item={'name': 'test_vlan1', 'id': 11})
changed: [tallinn-sw1] => (item={'name': 'test_vlan2', 'id': 12})
changed: [tallinn-sw1] => (item={'name': 'tallinn', 'id': 15})

TASK [conf_switches : Configure access interfaces]
************************************************
skipping: [tallinn-sw1]
changed: [tallinn-sw3]
```

```
changed: [tallinn-sw2]

TASK [conf_switches : Configure access interfaces]
*************************************************
skipping: [tallinn-sw2]
skipping: [tallinn-sw3]
changed: [tallinn-sw1]

TASK [conf_switches : Set SNMP community]
*********************************************************
changed: [tallinn-sw1]
changed: [tallinn-sw3]
changed: [tallinn-sw2]

TASK [conf_switches : Allow only snmp server]
*****************************************************
changed: [tallinn-sw1]
changed: [tallinn-sw3]
changed: [tallinn-sw2]

TASK [conf_switches : Disable HTTTP servers]
*****************************************************
changed: [tallinn-sw1]
changed: [tallinn-sw3]
changed: [tallinn-sw2]

TASK [conf_switches : set NTP server]
*************************************************************
changed: [tallinn-sw1]
changed: [tallinn-sw3]
changed: [tallinn-sw2]

TASK [conf_switches : Set con0 parameters]
*********************************************************
changed: [tallinn-sw1]
changed: [tallinn-sw3]
changed: [tallinn-sw2]

TASK [conf_switches : Set vty parameters]
*********************************************************
changed: [tallinn-sw1]
changed: [tallinn-sw3]
changed: [tallinn-sw2]

TASK [conf_switches : Save configuration]
*********************************************************
changed: [tallinn-sw1]
changed: [tallinn-sw3]
changed: [tallinn-sw2]

TASK [conf_switches : Print switch info]
*********************************************************
ok: [tallinn-sw1] => msg: Switch name=tallinn-sw1 location=tallinn serial_nr=XXXXX Uplink=24
ok: [tallinn-sw2] => msg: Switch name=tallinn-sw2 location=tallinn serial_nr=XXXXX Uplink=48
ok: [tallinn-sw3] => msg: Switch name=tallinn-sw3 location=tallinn serial_nr=XXXXX Uplink=24

PLAY RECAP
*********************************************************************************************
tallinn-sw1   : ok=13   changed=11   unreachable=0    failed=0    skipped=2    rescued=0
ignored=0
tallinn-sw2   : ok=13   changed=11   unreachable=0    failed=0    skipped=2    rescued=0
ignored=0
tallinn-sw3   : ok=13   changed=11   unreachable=0    failed=0    skipped=2    rescued=0
ignored
```

Lisa 5 – Väljavõte ühest konfigureeritava kommutaatori konfiguratsioonist

```
tallinn-sw1#show run
Building configuration...
version 15.2
no service pad
service timestamps debug datetime msec
service timestamps log datetime msec
service password-encryption
!
hostname tallinn-sw1
!
boot-start-marker
boot-end-marker
!
username admin privilege 15 password 7
046B0A14002E4000584B56
no aaa new-model
switch 1 provision ws-c2960x-24ts-l
!
ip domain-name example.com
vtp mode transparent
!
spanning-tree mode rapid-pvst
spanning-tree extend system-id
!
vlan 11
 name test1
!
vlan 12
 name test2
!
vlan 15
!
vlan 20
 name MGMT
!
interface FastEthernet0
 no ip address
 shutdown
!
```

```
interface GigabitEthernet1/0/1
 description tallinn
 switchport access vlan 15
 switchport mode access
 no cdp enable
 spanning-tree portfast edge
!
interface GigabitEthernet1/0/2
 description tallinn
 switchport access vlan 15
 switchport mode access
 no cdp enable
 spanning-tree portfast edge
!
interface GigabitEthernet1/0/3
 description tallinn
 switchport access vlan 15
 switchport mode access
 no cdp enable
 spanning-tree portfast edge
!
interface GigabitEthernet1/0/4
 description tallinn
 switchport access vlan 15
 switchport mode access
 no cdp enable
 spanning-tree portfast edge
!
interface GigabitEthernet1/0/5
 description tallinn
 switchport access vlan 15
 switchport mode access
 no cdp enable
 spanning-tree portfast edge
!
interface GigabitEthernet1/0/6
 description tallinn
 switchport access vlan 15
 switchport mode access
 no cdp enable
 spanning-tree portfast edge
!
interface GigabitEthernet1/0/7
 description tallinn
 switchport access vlan 15
 switchport mode access
 no cdp enable
```

```
 spanning-tree portfast edge
!
interface GigabitEthernet1/0/8
 description tallinn
 switchport access vlan 15
 switchport mode access
 no cdp enable
 spanning-tree portfast edge
!
interface GigabitEthernet1/0/9
 description tallinn
 switchport access vlan 15
 switchport mode access
 no cdp enable
 spanning-tree portfast edge
!
interface GigabitEthernet1/0/10
 description tallinn
 switchport access vlan 15
 switchport mode access
 no cdp enable
 spanning-tree portfast edge
!
interface GigabitEthernet1/0/11
 description tallinn
 switchport access vlan 15
 switchport mode access
 no cdp enable
 spanning-tree portfast edge
!
interface GigabitEthernet1/0/12
 description tallinn
 switchport access vlan 15
 switchport mode access
 no cdp enable
 spanning-tree portfast edge
!
interface GigabitEthernet1/0/13
 description tallinn
 switchport access vlan 15
 switchport mode access
 no cdp enable
 spanning-tree portfast edge
!
interface GigabitEthernet1/0/14
 description tallinn
 switchport access vlan 15
```

```
 switchport mode access
 no cdp enable
 spanning-tree portfast edge
!
interface GigabitEthernet1/0/15
 description tallinn
 switchport access vlan 15
 switchport mode access
 no cdp enable
 spanning-tree portfast edge
!
interface GigabitEthernet1/0/16
 description tallinn
 switchport access vlan 15
 switchport mode access
 no cdp enable
 spanning-tree portfast edge
!
interface GigabitEthernet1/0/17
 description tallinn
 switchport access vlan 15
 switchport mode access
 no cdp enable
 spanning-tree portfast edge
!
interface GigabitEthernet1/0/18
 description tallinn
 switchport access vlan 15
 switchport mode access
 no cdp enable
 spanning-tree portfast edge
!
interface GigabitEthernet1/0/19
 description tallinn
 switchport access vlan 15
 switchport mode access
 no cdp enable
 spanning-tree portfast edge
!
interface GigabitEthernet1/0/20
 description tallinn
 switchport access vlan 15
 switchport mode access
 no cdp enable
 spanning-tree portfast edge
!
interface GigabitEthernet1/0/21
```

```
 description tallinn
 switchport access vlan 15
 switchport mode access
 no cdp enable
 spanning-tree portfast edge
!
interface GigabitEthernet1/0/22
 description tallinn
 switchport access vlan 15
 switchport mode access
 no cdp enable
 spanning-tree portfast edge
!
interface GigabitEthernet1/0/23
 description tallinn
 switchport access vlan 15
 switchport mode access
 no cdp enable
 spanning-tree portfast edge
!
interface GigabitEthernet1/0/24
 switchport trunk allowed vlan 11,12,20
 switchport mode trunk
!
interface GigabitEthernet1/0/25
!
interface GigabitEthernet1/0/26
!
interface GigabitEthernet1/0/27
!
interface GigabitEthernet1/0/28
!
interface Vlan1
 ip address dhcp
!
interface Vlan55
 ip address 192.168.20.11 255.255.255.0
!
ip default-gateway 192.168.20.1
!
no ip http server
no ip http secure-server
ip ssh version 2
!
access-list 1 permit 192.168.0.20
!
snmp-server community test.community RO 1
```

```
snmp mib flash cache
!
line con 0
 logging synchronous
 stopbits 1
line vty 0 4
 exec-timeout 0 0
 logging synchronous
 login local
 transport input ssh
line vty 5 15
 exec-timeout 0 0
 logging synchronous
 login local
 transport input ssh
!
ntp server 192.168.0.21
!
end
```