TALLINNA TEHNIKAÜLIKOOL
Infotehnoloogia teaduskond

Georg Kahest 176009IDAR

Korratav valikaadressipõhine nimelahenduse teenus

diplomitöö

Juhendaja: Toomas Lepik
Magister

Tallinn 2021

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: Georg Kahest

17.05.2021

Annotatsioon

Antud diplomitöö eesmärgiks on luua töövahend, millega saab hõlpsalt juurutada ja valideerida valikaadressi põhist nimelahenduse teenust. Kusjuures see teenus võib paikneda nii füüsilistel, virtuaalsetel kui ka konteineritel põhinevatel serveritel.

Töö kätkeb endas mitmeid interneti alustehnoloogiaid ning nendega seonduvaid eelnevaid töid ja lahendusi, vaadeldes neid tööks vajalikus mahus.

Eelnevate teoreetiliste ning praktiliste tööde analüüsimise toel luuakse nõuded kogu lahendusele. Seejärel analüüsitakse neid, mille käigus tekivad eesmärgid ja nõuded, luuakse metoodika põhjal mudel, mille järgi juhinduda arendusprotsessis, ning rakendatakse mudelit nõnda, et tulemuseks on tööriist, millega on võimalik taaskorratavalt luua valikaadressipõhist nimelahenduse teenust.

Lõputöö on kirjutatud Eesti keeles ning sisaldab teksti 35 leheküljel, 8 peatükki, 14 joonist.

Abstract

Reproducible Anycast DNS Service

The thesis goal is to design a solution for deploying and managing anycast based DNS service where the service nodes could be on bare metal, virtual machines or containers.

To achieve that goal technological stack was studied and explained in a extent to cover all aspects critical for understanding the technology.

That included showing how DNS works and high level overview of BGP what was needed for completeness. Prior works related to DNS and anycast were studied in a extent to formulate requirements for the solution and the tooling required for completing the task.

Based on the requirements, tooling was chosen and development plan was formulated using methodology defined in the paper. The development plan included breaking the problem into smaller iterations with their own goals, requirements and validation criterias.

Development process included building simulation environment where all the functionality was tested and validated against the predefined targets, the simulation environment tooling was chosen with requirement in mind that hardware platforms could be included in the testing if support for them was developed in the future.

The practical result of the work is Ansible role what allows management of Bird routing demon and Bind name server demon to form anycast DNS service with several routers and service nodes to provide high availability and allows to test that functionality with the role itself with that, validating the functionality of the solution.

The thesis is in Estonian and contains 35 pages of text, 8 chapters, 14 figures.

Lühendite ja mõistete sõnastik

API	*Application Programming Interface,* Masinliides
AS	*Autonomus System*, Autonoomne süsteem
ASN	*Autonomus System Number*, Autonoomse süsteemi number
BFD	*Bi-directional Forwarding Detection,* Kahepoolne suunamise tuvastus
BGP	*Border Gateway Protocol*, Piirilüüsi protokoll - marsruutimise protokoll
CNAME	*Canonincal Name,* Viitav nimi
DevOPS	*Development and Operations,* Arendus ja haldus
DNS	*Domain Name System*, Domeeninimesüsteem
ECMP	*Equal-Cost Multi Path,* Mitme võrdse marsruudi kasutamine
IP	*Internet Protocol*, Interneti protokoll
IPTV	*Internet Protocol Television,* Interneti televisiooni protokoll
NS	*Name Server*, Nimeserver
OSI	*Open Systems Interconnection*, Avatud süsteemide kokkuühendamine
OSPF	*Open Shorthest Path First, A*vatud lühim tee esimesena, marsruutimie protokoll
PIAS	*Proxy IP Anycast Service,* Vaheserveri IP valiksaadressi teenus
RFC	*Request for Comment*, Kutse ettepanekuteks
SRE	*Site Reliability Engineer*, Käideldavuse insener
SSH	*Secure Shell Protocol,* Turvalise kaughalduse protokoll
TCP	*Transmission Control Protocol*, Transmissiooni protokoll
TLD	*Top Level Domain,* Tipp taseme domeen
uCARP	*Userland Common Address Redundancy Protocol,* ühise aadressi protokoll
UDP	*User Datagram Protocol*, Kasutajadatagrammi protokoll
VRRP	*Virtual Router Redundancy Protocol,* Virtuaalse ruuteri liiasuse protokoll
WWW	*World Wide Web,* Ülemaailmne võrgend

Sisukord

1 Sissejuhatus....9
1.1 Probleem....10
1.2 Eesmärk....10
1.3 Metoodika....11
1.4 Ülevaade tööst....12
2 Alustehnoloogiad ja taust....13
2.1 IP marsruutimine....13
2.1.1 Monosaade....13
2.1.2 Levisaade....14
2.1.3 Multisaade....14
2.1.4 Valiksaade....14
2.1.5 IP-Prefiks ja vaikemarsruut....15
2.2 Domeeninimesüsteem....15
2.3 BGP....16
2.3.1 BGP liigid....18
2.3.2 BFD ja ECMP....19
2.4 Valikaaddress....19
2.5 Nimeteenus .ee domeeniregistris....20
2.5.1 Esimesed klastrid....21
2.5.2 Keerukuse kasv....21
2.6 Varasemad tööd....22
2.6.1 Teoreetilised....23
2.6.2 Konfiguratsiooni haldus vahendite moodulid....25
2.6.3 Konfiguratsiooni generaatorid (BIRD seadistajad)....25
2.6.4 Terraform....26
3 Nõuded ja eesmärgid....27
3.1 Nõuded loodavale lahendusele....27
3.2 Nõuded valitavale tarkvarakomplektile....27

3.2.1 Eesmärgid....28
3.2.2 Nõuded....28
4 Arenduse iteratsioonid....29
4.1 Iteratsioonide mudel....29
4.2 Iteratsioon BIRDS....30
4.2.1 Eesmärgid....30
4.2.2 Nõuded....30
4.3 Iteratsioon NIMED....30
4.3.1 Eesmärgid....30
4.3.2 Nõuded....30
5 Realisatsioon....32
5.1 Tarkvarakomplekti valimine....32
5.1.1 Konfiguratsioonihaldus vahend....32
5.1.2 Testimis/valideerimise lahendus....33
5.2 Iteratsioon BIRDS....35
5.2.1 Rakendamine....35
5.2.2 Lisandunud omadused/funktsionaalsus....35
5.3 Iteratsioon NIMED....36
5.3.1 Rakendamine....36
5.3.2 Lisandunud omadused/funktsionaalsus....37
6 Valideerimine ja tulemused....38
6.1 Iteratsiooni BIRDS valideerimine....38
6.1.1 Valideerimise tingimused....38
6.1.2 Realisatsioon....39
6.2 Iteratsiooni NIMED valideerimine....39
6.2.1 Valideerimise tingimused....39
6.2.2 Realisatsioon....39
6.3 Tulemused....40
7 Lahenduse laiendamine....42
8 Kokkuvõte....43
Kasutatud kirjandus....44
Lisa 1– Lihtlitsents lõputöö reprodutseerimiseks ja lõputöö üldsusele kättesaadavaks tegemiseks....46

Jooniste loetelu

Joonis 1.Monosaade [60].....13
Joonis 2.Levisaade [60].....14
Joonis 3.Multisaade[60].....14
Joonis 4.Valiksaade[60].....14
Joonis 5.DNS nimepäring.....16
Joonis 6.Väline BGP.....18
Joonis 7.Sisene BGP.....18
Joonis 8.Valikaadressi teenus.....20
Joonis 9.Lahenduse ülevaatlik skeem.....32
Joonis 10.Google Trends Ansible.....33
Joonis 11.Testimise lahenduse skeem.....34
Joonis 12.BIRDS skeem.....35
Joonis 13.Nimeserveri juhtimise skeem.....36
Joonis 14.Realiseeritud lahenduse skeem.....40

1 Sissejuhatus

Domeeninimesüsteem (edasipidi nimeteenus / DNS - ing.k *Domain Name System*) on üks olulisema tähtsusega interneti infrastruktuuri teenuseid, tänu millele me saame interneti ressursside nagu näiteks WWW (ing.k *World Wide Web*) poole pöördumiseks kasutada sõnu (nimesid) numbrite (IP-aadresside) asemel.

Antud töö kirjutamise ning realiseerimise inspiratsioon tuli autori pikaajalisest kogemusest selle teenuse käitamisega Eesti .EE riigi tunnuskoodiga domeeniregistris.

Kasutajate ootused .EE nimeteenuse käideldavusele ning aina sagenevad ründed DNS-infrastruktuuri vastu sundisid otsima mooduseid, kuidas parandada teenuse kättesaadavust, ning muutma teda rünnakute suhtes vastupidavamaks.

Valikaadressipõhine nimelahenduse teenus on väga levinud põhiliselt otseselt nimeteenusega seotud infrastruktuurides, aga asjakohane on sellist lahendust kasutada igas keskkonnas, kus on kasutusel dünaamiline marsruutingu (edaspidi „ruuting“) kiht.

Tema ülesehitamise suhteline keerukus, ent siiski väga hea korratavus muudavad ta heaks kandidaadiks, mille juurutamist automatiseerida ning seeläbi lihtsustada ning standardiseerida lahenduse juurutamist. Lähtudes sellest, hakkaski autor planeerima sellise tööriista loomist, mis seda võimaldaks.

Töö eesmärgiks on luua tööriist, mis võimaldaks taaskorratavalt luua valikaadressi põhist nimelahenduse teenust.

Analüüsides eelnevaid töid ning arvestades parimate praktikatega, plaanitakse luua nende põhjal lahenduse nõuded ja eesmärgid. Loodava tööriista arendusprotsessis kasutatakse meetodit, kus jagatakse kogu ülesanne väiksemateks lahendatavateks osadeks ja neid järge mööda lahendades jõutakse lõpuks kogu nõutud funktsionaalsuseni.

1.1 Probleem

Nimeteenuse puhul on kasutatud valikaadressi juba aastakümneid produktsioonis kõige kriitilisemas kohas, juurtsooni nimeserverites. Tänaseks peitub 13 juurnimeserveri taha, mida haldab 12 erinevat organisatsiooni, lausa 1378 serverit. [1]

Lisaks juurnimeserveritele on valikaddressipõhine nimeteenus laialt levinud ka TLD (ing.k Top Level Domain) operaatorite seas aga asjakohane on teda kasutada igal pool kus on kõrgendatud käideldavuse nõuded ja soovitakse nimeserverid paigutada mitmesse asukohta. Vajaliku tööriista olemas olul võiks seda kasutada ka suuremates ettevõtetes rekursiivse nimeteenuse käideldavuse suurendamiseks.

Hoolimata sellest, ei ole olemas ühte tööriista, mis võimaldaks sellist teenust luua. Peamiselt selle pärast, et sellise teenuse seadistamine on väga keskkonnaspetsiifiline ülesanne ning nõuab mitmete tarkvara komponentide seadistamist [2] , mis kogumis on suhteliselt keeruline ülesanne.

Ajalooliselt on sellise teenuse loomisel olnud takistuseks ka rollid organisatsioonides: tihtipeale on võrgukihi ja teenuste haldus kuulunud erinevate inimeste pädevusse, tänapäeval on see trend aga muutumas seoses DevOps (ing.k Development and Operations) ja SRE (ing.k *Site Reliability Engineer*) mõttelaadi levikuga.

Johtuvalt teenuste käideldavuse nõuete ja seadistuse keerukuse kasvust, administreerimise meetodite arenemisest ning tarkvara ja raudvara lahenduste muutumisest jõuamegi töös käsitletava probleemini: taaskorratava valikaadressi põhise nimelahenduse teenuse loomine, mis võib paikneda füüsilistel, virtuaalsetel või konteineritel põhinevatel serveritel.

1.2 Eesmärk

Töö eesmärgiks on kaardistada vajalikud tehnoloogiad ning tutvuda varasemate teoreetiliste ning praktiliste töödega nimeteenuse ning valikaadressi põhiste lahenduste osas.

Tuleb komponeerida tööriist, millega saab luua, seadistada ning valideerida valikaadressi põhist nimeserveriteenust. Seda läbi marsruuterite ning Linuxil põhinevate teenusserverite seadistamise.

Väga oluliseks on lahenduse funktsionaalsus oma toimimise valideerimiseks ning modulaarne disain, mis lubaks tulevikus kõiki tarkvara juhtimise komponente välja vahetada.

1.3 Metoodika

Kasutan töös enda jaoks kohandatud disainiteaduse metoodikat [3] koos simulatsioonidega [4], mille kaudu valideerin, et iga iteratsioon on oma eesmärgid täitnud. Sarnast metoodikat on kasutatud infotehnoloogiliste probleemide lahedamiseks juba enam kui 15 aastat. Hoolimata oma vanusest, on antud metoodika kasutamine siiski aktuaalne [5].

Esmalt vaatleme alustehnoloogiad ning analüüsime eelnevaid töid, millega töötades tekivad nõuded ning eesmärgid lahendusele. Kasutades kohandatud metoodikat, jagatakse arendustöö iteratsioonideks millele on võimalik luua valideerimise tingimused, ja seejärel arendan nõutud funktsionaalsuse loodavas tööriistas.

Üldiselt koosneb antud metoodika kuuest sammust, mida reeglina alustatakse esimesest, aga miski ei keela alustada ka mõnest kaugemast sammust, kui lahendus seda nõuab.

Metoodika soovitab uurimus küsimuse jagada piisavalt väikesteks osadeks, et oleks võimalik ühe iteratsiooniga üks konkreetne probleem/osa ära lahendada ja seejärel võib liikuda järgmise küsimuse juurde, kuni lõpuks on kogu keerukus kaetud.

1. Probleemi tuvastamine ja motivatsioon: tööriista puudumine valikaadressi teenuse loomiseks.

2. Lahenduse tulemused: tööriist artefakt, mis lahendust realiseerib.

3. Disain ja arendus: läbi nõuete ning valideerimistingimuste leidmise ning nende realiseerimise iteratsioonide kaudu.

4. Demonstratsioon: läbi valideerimise simulatsiooni keskkonnas.

5. Hindamine: valideerimise edukus.

Kuna tulemuseks olev tööriist peab olema ka reaalelus kasutatav, siis tuleb lähtuda tema loomisel printsiipidest, mida tänapäevases tarkvaraarenduses rakendatakse ehk kogu lahendus peab olema ka testitav.

Edukaks võib tulemust lugeda siis, kui lahendus suudab hallata valikaadressi teenust vastavalt nõuetele ning see on valideeritud simulatsioonide kaudu.

Piirangud: lahendus ei halda nimeteenuse sisu vaid kasutab eelloodud tsoonifaili lahenduse valideerimiseks.

Alustehnoloogiaid käsitletakse piiratud skoobis, puudutades nendest rääkides ainult antud lahendusele vajalikke osi, kui just nendest rääkimine ei aita paremini laiemat pilti avada.

1.4 Ülevaade tööst

Esimeses peatükis juhatame teema sisse, näitame kust tuleb probleem, millised on eesmärgid ning millise metoodikaga me seda lahendama hakkame.

Teises peatükis vaatleme tehnoloogilist tausta ning vaatleme eelnevaid töid ja lahendusi millega täna oleks võimalik säärast teenust luua ning analüüsime neid. Lisaks tutvume .EE domeeniregistri nimeteenuse arenguga kus ilmnes kõige teravam vajadus taolise tööriista järgi.

Kolmandas peatükis juhindudes taustast ja varasematest töödest loome lahendusele nõuded ja eesmärgid.

Neljandas peatükis tutvustame arendus mudelit ja rakendades seda jagame töö iteratsioonideks millel on oma eesmärgid ja nõudmised.

Viiendas peatükis lahendame analüüsi teel nõutud tingimused ning realiseerime eesmärgid ja nõuded tarkvaralise lahendusena.

Kuuendas peatükis valideerime tulemusi ja anname hinnangu lahenduse realisatsioonile.

Seitsmendas peatükis pakume välja kuidas lahendust laiendada.

Kaheksas peatükk on kokkuvõte.

2 Alustehnoloogiad ja taust

Käesolevas peatükis tutvutakse esmalt tehnoloogilise baasiga, millele antud töö toetub. Selleks, et eestikeelsel lugejal oleks võimalik tööd paremini mõista, olenemata nende eelnevast taustast, peab avama vähemalt pinnapealselt mõned infotehnoloogiaga ja spetsiifiliselt arvutivõrkudega igapäevaselt kokkupuutuvatele inimestele enesestmõistetavad tehnoloogiad ka emakeeles.

Lisaks antakse ülevaade .ee domeeniregistri nimeserveri teenusest ja tema arengust ning tehnilistest valikutest, kuna need olid otseseks tõukeks antud töö kirjutamiseks.

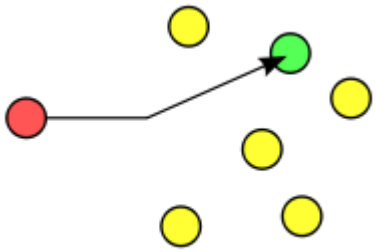
2.1 IP marsruutimine

IP (ing.k *Internet Protocol*), mis on, nagu ka nimi viitab, meie interneti peamiseks protokolliks, võib lausa öelda üheks põhiliseks sambaks. Välja pakuti ta juba 1981. a septembris ja baseerus ta vanematele protokollidele. Tema eesmärgiks oli luua protokoll, mille abil oleks võimalik erinevaid arvutivõrke kokku ühendada ehk ta oli eelduseks interneti loomisele, mis ongi oma definitsiooni järgi võrkude võrk. [6]

Selleks, et paremini mõista valiksaadet, peame tutvuma teiste võimalike saatemeetoditega.

2.1.1 Monosaade

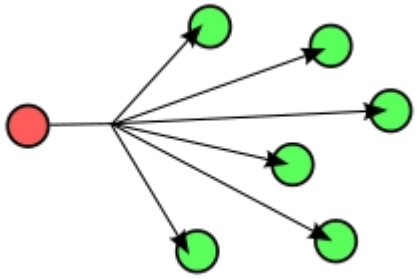
Kõige tavalisem suhtlus arvutivõrkudes toimub monosaate (ing.k *unicast*) kaudu, mis on oma olemuselt üks-ühele suhtlus. Joonisel näeme, et punane täpp suhtleb rohelisega, ilma et ükski kollane oleks liiklusesse kaasatud. Võimaldab kasutada nii TCP-[7] kui ka UDP-põhist[8] kohaleviimist.



Joonis 1.Monosaade [60]

2.1.2 Levisaade

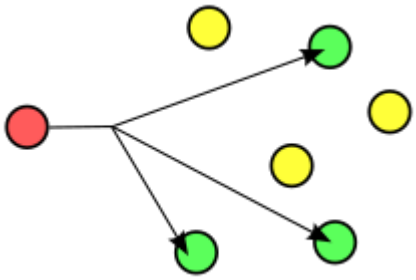
Levisaade pakuti väljajuba 1984. aastal ja on oma olemuselt üks-mitmele saatmise tüüp. [9] Võimaldab kasutada ainult UDP-põhist suhtlust. Teenuse näiteks võiks olla mõni sisevõrgus töötav teenus mille olemasolu reklaamitakse antud võrgu segmendile. Joonisel on näha, et punaselt täpilt välja saadetud info jõuab kõigi rohelisteni.



Joonis 2.Levisaade [60]

2.1.3 Multisaade

Multisaade on oma olemuselt kas üks mitmele või mitu-mitmele saatmise tüüp, aga peamiselt siiski üks mitmele.[10] Peamiseks kasutusalaks on meedia saatmine ühe operaatori võrgu piires, näiteks IPTV (ing.k Internet Protocol television) näol. [11]

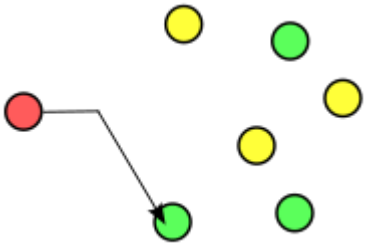


Joonisel näeme, et punane täpp suhtleb rohelistega (nad kõik on liitunud multisaate grupiga) ja kollased ei ole suhtlusesse kaasatud kuigi asuvad samas võrgus.

Joonis 3.Multisaade[60]

2.1.4 Valiksaade

Valiksaade pakuti välja aastal 1993, on oma olemuselt üks-ühele suhtlusskeem, aga seda erisusega, et partner võib ajas muutuda. Algseks kasutusalaks oli DNS aga tänapäeval ka teised protokollid. On võimalik kasutada nii TCP- kui UDP-põhist kohaleviimist olenevalt kuidas kogu lahendus disainida. [12]



Joonisel näeme, et punane täpp suhtleb rohelisega, aga võib juba järgmise ühendusega sattuda suhtlema järgmise rohelisega.

Joonis 4.Valiksaade[60]

2.1.5 IP-Prefiks ja vaikemarsruut

IP-prefiks on IP aadressi kujuline konstruktsioon kuhu on lisatud alamvõrgumask bit vormingus, näiteks 192.168.53.1/24 antud juhul on alamvõrguks 192.168.53.0, IP aadressiks 192.168.53.1 ning võrgumaskiks /24 ehk 255.255.255.0 [13] .

Vaikemarsruut on IP-prefiks, kuhu saadetakse kogu liiklus, millele puuduvad täpsemad marsruudid. Teda kirjeldatakse unikaalse IP-prefiksiga 0.0.0.0/0 mis tähistab kogu adresseeritavat IP-võrku. Kasutatakse teda praktiliselt igas seadmes, mis võrgus on, ja viitab ta reeglina kohaliku võrgu vaikelüüsile.

2.2 Domeeninimesüsteem

Domeeninimeteenus on üks varasemaid teenuseid, mida arvutivõrkude arenedes inimesed vajasid. Kuna oma loomult ei ole inimesed head numbrite meelde jätmises, on igati loogiline, et vajati teenust, mille kaudu transleeritakse nimed IP-aadressideks, ja inimesed hakkasid peamiselt kasutama nimesid, et jõuda võrgu ressurssideni.

Formaalselt loetakse DNS protokolli alguseks RFC1034 [14] ning RFC1035 [15], mille loomisest on tänaseks päevaks möödunud juba enam kui 33 aastat. Selle aja jooksul on loodud üle 270 RFC (ing.k *Request For Comment*), mis protokolli laiendavad ja täiendavad [16], mis on juba iseenesest kõnekas fakt, mis annab aimu, kui oluline see teenus interneti toimimisel on.

Kasutusel on erinevad kirjetüübid, igaüks omab oma spetsiifilist eesmärki ja kasutuskohta, aga antud töö raames on olulised ainult järgmised:

- NS (ing.k *Nameserver*) ehk siis nimeserveri kirjed, mis viitavad kus antud tsooni nimeserverid asuvad.
- A kirje võimaldab lahendada nimesid IP-aadressideks.
- CNAME (ing.k *Canonical Name*) võimaldab ühe nime panna viitama teisele nimele.

Iga tsoon ehk siis üks tase (näiteks .ee) koosneb paljudest erinevatest kirjetest, mis võivad viidata samasse tsooni või mõnda teise.

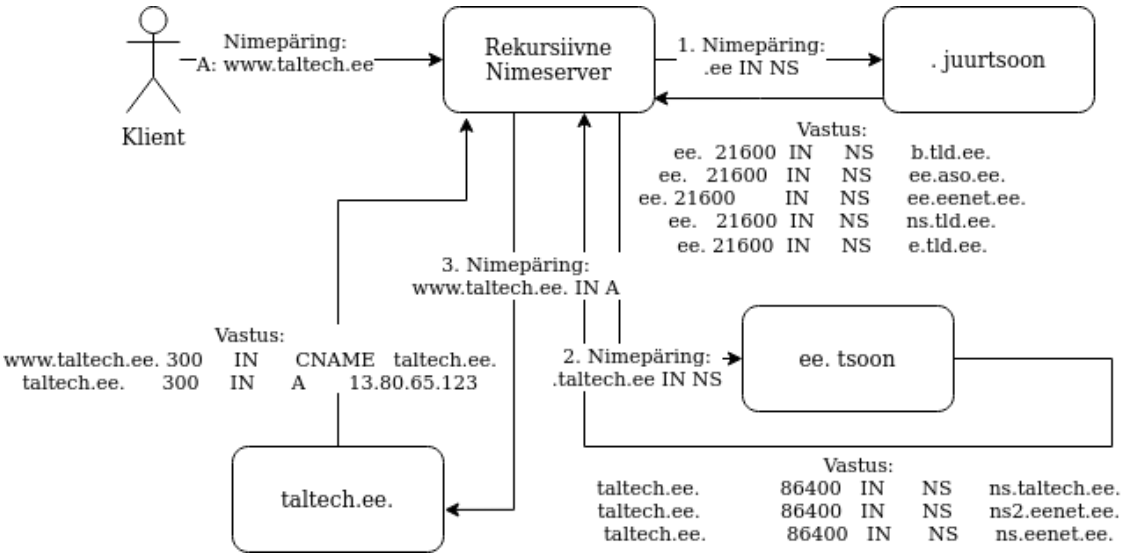
Nimeserverid jagunevad oma rolli järgi kaheks:

- Autoriteetsed – omavad andmeid ühe või rohkema tsooni kohta.
- Rekursiivsed – lahendavad nimesi.

DNS on hierarhiline süsteem (puu), kus iga tase (puu oks) on eraldiseisev osa (tsoon), millele viitab ülemine tase ja mis võib viidata endast järgmisele tasemele, ainukeseks erandiks on puu tipp ehk juurtsoon.

Viited tema NS kirjetele on sisse kirjutatud rekursiivset nimelahendamist võimaldavate tarkvarade seadistusse või lähtekoodi. Kõik ülejäänud andmed kogutakse läbi viidete ühelt tasemelt teisele.

Kõige lihtsam on mõista DNS päringut läbi näidise, mis asub järgneval joonisel:



Joonis 5.DNS nimepäring

Klient esitab päringu oma rekursiivsele päringule, kes alustab nimelahendamist juurtsoonist, kust saab vastuse selle kohta, kust võib leida .ee tsooni omavad nimeserverid, kellelt omakorda küsitakse, kus asuvad taltech.ee nimeserverid, kellelt omakorda küsitakse algne küsimus, kus asub www.taltech.ee, ja saadakse vastuseks CNAME kirje, mis viitab taltech.ee domeenile, mille A kirje 13.80.65.123 on aadress, mis antakse vastusena tagasi kasutajale.

2.3 BGP

Pole just lihtne leida protokolle, mis oleks interneti toimimisel kriitilisemal kohal kui DNS. Siiski on vähemalt üks selline olemas ja selleks on BGP (*Border Gateway Protocol*) [17]

spetsifitseeritud aastal 1989 ja tema ülessandeks on AS-ide (ing.k *Autonomus System*) vahel ruutingu informatsiooni vahetamise võimaldamine.

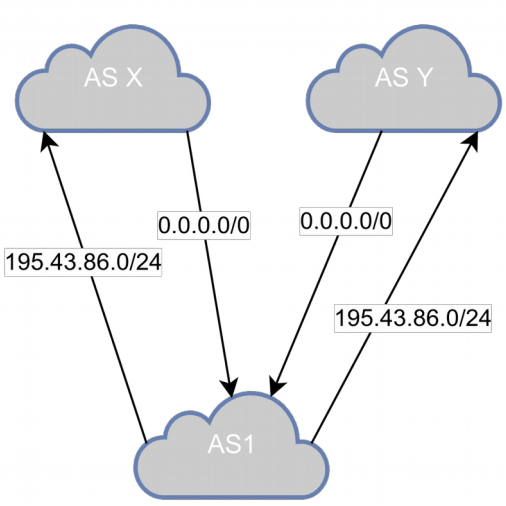
AS-iks loetakse tavaliselt ühte autonoomset võrku või organisatsiooni, kes võrku haldab. AS-i tuvastamine toimub läbi ASN-i (*Autonomus System Number*) - see on unikaalne number terve interneti piires, mis saadakse omale kohalikust internetiregistrist, kelleks näiteks Euroopas on RIPE NCC[18]. Siiski on see number unikaalne internetiüleselt, st erinevad internetiregistrid kasutavad erinevaid vahemikke, et mitte konflikti sattuda. [19]

Lisaks on defineeritud privaatne ASN-vahemik, millest iga organisatsioon võib kasutada suvalisi numbreid oma sisemises infrastruktuuris BGP-ühenduste loomiseks ning loogiliselt erinevate osade defineerimiseks. [20]

Informatsioon mida võrkude vahel vahetatakse, koosneb IP-prefiksist ning ruuteri IP-aadressist, kelle kaudu sinna pääseb. Reeglina on selleks ruuteri IP, kellega BGP sessioon on loodud, aga põhimõtteliselt on võimalik, et seal on ka mõni muu ruuteri IP, näiteks kui on kasutusel marsruutide peegeldamine [21].

2.3.1 BGP liigid

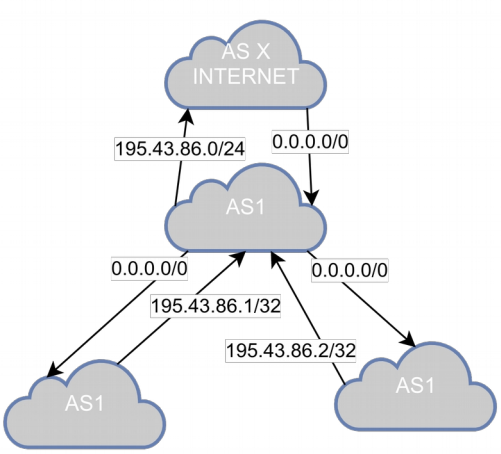
BGP jaguneb kahte liiki, kõige tavalisemaks kasutusjuhuks on väline BGP, mida iseloomustab järgnev joonis.



Joonis 6.Väline BGP

AS1 ostab teenust AS X ja AS Y käest, reklaamib (teavitab) neile võrke, mis asuvad tema AS-is või ühenduvad tema külge (195.43.86.0/24). Nemad aga saadavad AS1-le erinevaid IP-prefikseid ning vaikemarsuudi ehk lubavad, et nende kaudu pääseb kõikidesse võrkudesse.

Teiseks liigiks on sisemine BGP, mille puhul BGP-ühendused on loodud ühte AS-i kuuluvate ruuterite vahel, näiteks ühe ettevõtte erinevates andmekeskustes olevad marsruuterid võivad seeläbi vahetada informatsiooni teadaolevate võrkude kohta. Järgneval joonisel on näha, kus kolme AS1 ruuteri vahel on sisemine BGP-ühendus ja üks nendest ühendab nad kõik välisvõrgu külge.



Joonis 7.Sisene BGP

Samuti võimaldab sisemine BGP luua ühe AS-i piires valikaadressi teenuseid.

2.3.2 BFD ja ECMP

BFD (ing.k *Bidirectional Forwarding Detection*) aitab kiiremini reageerida partneri tõrke tekkimisel ja seeläbi kiiremini BGP-sessiooni sulgeda, kui seda ainult BGP protokolli kasutamine võimaldaks. [22]

ECMP (ing.k *Equal-Cost Multipath*) aitab suurendada läbilaset, kasutades mitut samaväärset teed (marsruuti) järgmise ruuteri/teenus serverini [23].

2.4 Valikaaddress

IP valikaadress pakuti formaalselt välja juba 1993.a aastal RFC1546 [24] dokumendiga. Juba tollal nähti väärtust selles, et saab kasutajaid juhatada lähima teenust pakkuva serverini ja nõnda lihtsustada teenuste leidmist/tuvastamist ning et see annab võimaluse luua dubleeritud teenuseid ning seeläbi võimaldab kõrgkäideldavate teenuste loomist.

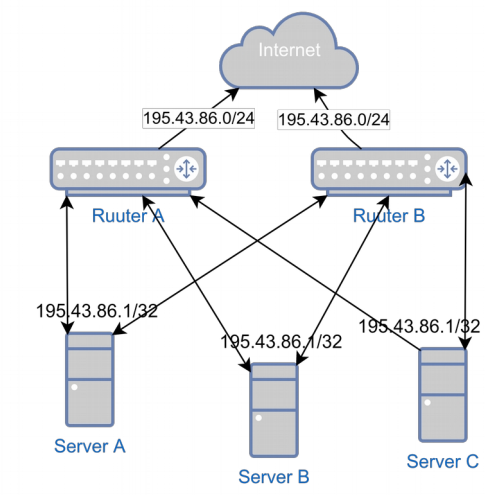
Juba algses dokumendis tuuakse välja ka peamised probleemid, mis taolise teenuse loomisel tekivad tulenevalt UDP ja TCP erinevustest. Nimelt on ajalooliselt soovitatud valikaadressi kasutada pigem UDP-põhiste teenuste puhul, kuna UDP on olekuvaba protokoll, seega ei ole probleemiks, kui üks klient satub mitut erinevat teenusserverit kasutama [2].

Siiski on tänapäevaks mõeldud välja juba rida meetmeid, kuidas saavutada olukord, et valikaadress oleks kasutatav ka TCP-põhiste teenustega, seda peaasjalikult koormusjaoturite kihti kasutades enne teenusservereid, mis tagavad, et ühendused jõuavad sama teenusserverini.

Alternatiiviks oleks kasutada ruuterites selliseid koormusjaotuse räsialgoritme, mis on realiseeritud nõnda, et tagatakse ühenduste jõudmine determineeritult teenusserveriteni. [25]

Kuigi valikaadress pole definitsiooni järgi piiratud ainult BGP-ga ja varasema kirjanduse järgi võib teda samahästi rakendada ka OSPF-iga (ing.k *Open Shortest Path First*), on siiski tänapäeval standardiks pigem BGP kasutamine.

Järgneval joonisel on näide valikaadressi teenusest:



Joonis 8.Valikaadressi teenus

Tagamaks, et teenusserveri vea järgselt toimuks ümberlülitamine võimalikult kiiresti, tuleb rakendada protokolli BFD, lisaks peab teenusserver kontrollima enda töökorras olekut ja võimalikult kiiresti tuvastama, kui teenus ei tööta korrektselt, tagamaks seda, et kliendid ei jõua katkise teenusserverini. Lisaks saab suurendada teenuse läbilaskevõimet kasutades ECMP protokolli. [26]

2.5 Nimeteenus .ee domeeniregistris

Oma enam kui 10a kogemuse jooksul .ee domeeniregistris süsteemiadministraatorina oli autoril võimalus näha, kuidas arenesid meetodid ning kuidas teenuseid ja ka nimeteenust loodi ja käideldi.

Esialgselt paiknesid nimeserverid üksikutel serveritel. Selleks, et tagada soovitud käideldavuse tase, peab teenust tagama vähemalt kaks serverit, nõnda on võimalik paigaldada tarkvara- ja püsivarauuendusi ilma teenuse käideldavust kahjustamata.

Järgmiseks arenguetapiks oli teenindavate serverite paigutamine mitmesse geograafillisse asukohta, et tagada andmekeskuse tõrke korral teenuse kättesaadavus.

Lisaks hakkasid parimad praktikad rääkima sellest, et nimeserveri teenust peaksid tagama erinevad nimeserveritarkvarad, et vältida olukorda, et ühes tarkvaras leitud viga saab saatuslikuks kogu teenusele [2]. Sellest tekkis vajadus toetada ka virtuaalseid servereid, et oleks võimalik optimaalsemalt raudavara ära kasutada.

2.5.1 Esimesed klastrid

Esimeseks kõrgkäideldavust pakkuvaks lahenduseks oli VRRP (ing.k *Virtual Router Redundancy Protocol*), mis on suhteliselt levinud lahendus vaikelüüsi käideldavuse suurendamiseks, aga mitte väga levinud lahendus teenusserverite käideldavuse suurendamiseks. [27]

VRRP protokolli realiseeris uCARP [28], millega on võimalik luua jagatud virtuaalne IP-aadress. Kõik osalevad serverid reklaamivad enda olemasolu levisaate piirkonnas ja kõige kõrgema prioriteediga server võtab virtuaalse teenuse IP enda kasutusse. Seda prioriteeti muutes või kui kõrgeima prioriteediga serveril tekib tõrge, toimub automaatne ümberlülitamine järgmisele.

Kuna aga VRRP protokoll töötab OSI (ing.k Open Systems Interconnection) teises (võrgustikukihis) ning kolmandas (transpordi) kihis, siis eeldab ta, et serverid, mis IP-aadressi jagavad, asuvad samas levisaatealas. See aga seab lahenduse rakendamisele piirangu, et sisuliselt peavad serverid asuma samas andmekeskuses või siis peab andmekeskuste vahel olema kasutusel mõni võrgutunneli protokoll, mis aga lisab keerukust ning laiendab tõrkepiirkonda (ing.k *Failure Domain*) mõlemasse andmekeskusesse.

Üheks peamiseks probleemiks on aga see, et korraga saab kliente teenindada ainult üks server klastris, mis muudab lahenduse küllaltki ebaoptimaalseks, kui soov on omada rohkemaid aktiivseid teenusservereid.

2.5.2 Keerukuse kasv

Eelnevalt olid kasutusel VRRP klastrid, kus iga klaster omas kahte serverit ning mõlemad kasutasid ühte ja sama operatsioonisüsteemi:

- RedHat Enterprise Linux klaster
- Debian GNU/Linux klaster
- Solaris klaster

Kõik serverid kasutasid ühte ja sama nimeserveri tarkvara (BIND) see aga tähendas , et ühe kriitilise vea puhul selles tarkvaras oli kogu teenus haavatav.

Pikaajalise valikaadressi põhise nimeteenuse kasutajad töötasid välja parimate praktikate dokumendi, mis sätestab, et operatsioonisüsteemi ja rakendustarkvara vigadest tekkiva käideldavuse riski maandamiseks tuleb kasutada erinevaid tarkvarasid, mis aga teistpidi suurendab teenuse juurutamise ning haldamise keerukust [29].

Selle eesmärgi täitmiseks oli vaja kasutusele võtta lisaks BIND-ile ka Knot ning PowerDNS. Kui varasemalt olid kõik nimeserverid asunud otse raudvarale paigaldatud serverites, siis nüüd hakkas tekkima vajadus kasutada ka virtualiseeritud servereid, et oleks võimalik ühel serveril kasutata mitut erinevat nimeserveri tarkvara.

Arvestades VRRP puudusi, siis siinkohal jäigi ainukeseks mõistlikuks valikuks minna üle valikaadressi põhisele nimeserveri teenusele, mis aga tähendas, et teenusserverites pidi edaspidi kasutusel olema ka dünaamilise ruutingu protokoll. Lähtudes samast loogikast mis nimeserveri tarkvara puhul, võeti kasutusele Quagga ja BIRD.

Lisaks teenusserveritele pidi nüüd iga nimeserveri lisamisel seadistama ka ruutereid, mis kasutasid nii IOS-i kui ka Junost.

Sellise tehnoloogilise mitmekesisuse haldamiseks pidi kasutusele võtma konfiguratsioonihaldusvahendid, mida tehtigi etapiviisiliselt iga komponenti eraldi hallates.

Üsna pea selgus, et tegelikult tuleks sellist teenust hallata tervikuna, mitte üksikute komponentide kogumina. Aga kuna organisatsioonil oli ajakriitilisem automatiseerida tarkvara juurutamise protsessi, kasutades konteinertehnoloogiat, jäigi sellise tervikliku valikaadress põhise nimeteenuse haldamise vahend loomata.

2.6 Varasemad tööd

Kui teoreetilisi materjale on valikaadressi teenuse kohta võimalik leida suhteliselt palju, siis praktilisi tööriistu, millega seda realiseerida, napib. See tuleneb ilmselt sellest, et kõik, kes antud teenust kasutavad, on suured organisatsioonid, kellel on võimekus tööriistad majasiseselt luua. Reeglina tekivad nad orgaaniliselt infrastruktuuri haldusvahendite arengu käigus ehk tavaliselt luuakse lahendus kasutusel olevate konfiguratsioonivahendi tükkide sidumise teel.

Selline areng on igati loomulik ja mõistlik, aga harva sünnib sellise lähenemise teel tööriist, mida saaks ka väljaspool seda organisatsiooni kasutada, kui see just ei ole seatud eesmärgiks

eraldi. Mida vahendite puudumine näitabki, on see, et seni keegi veel sellist eesmärki seadnud ei ole.

Kuigi ei ole olemas terviklikku vahendit, millega sellist teenust luua, on siiski olemas hulk lahendusi mingite osiste seadistamiseks sellest lahendusest ja mõnda neist siinkohal ka vaadeldakse.

2.6.1 Teoreetilised

"RFC 3258 - Distributing Authoritative Name Servers via Shared Unicast Addresses" [30]

Töö kirjeldab riske, mis tekivad seoses ühe aadressi taha mitme teenusserveri paigutamisega.

Peamised järeldused, mis antud tööst teha võib, on:

- Suurem serverite arv tõstab seadistamise keerukust ja vigade tekkimise võimalust.
- Erinevate serverite sisu erinevus võib tekitada tõrkeid klientidele.
- Tsoonifaili distributeerimine muutub kriitiliseks protsessiks.
- Sellise lahenduse ründamine muutub raskemaks, kuna servereid on rohkem.

„A Software Approach to Distributing Requests for DNS Service using GNU Zebra, ISC BIND 9 and FreeBSD" [2]

Kirjeldab, kuidas luua valikaadressi põhist nimeserveri teenust, kusjuures väljapakutud lahendus on valideeritud F juurserveriga. Tegemist on ilmselt esimese dokumendiga, mis kirjeldab ära kogu vajaliku tarkvara komplekti valikaadressi teenuse loomiseks. Lisaks on autorid toonud välja rea nõudeid/tingimusi, millele toetutakse antud töös tugevalt oma lahenduse väljatöötamisel.

Peamiseks puuduseks saab lugeda ehk ainult seda, et töö sisu ei ole võimalik lihtsalt valideerida, millest tuleneb ka tingimus, et uus loodav lahendus peab olema lihtsalt valideeritav.

„Operations of Anycast Services „ [29]

Tegemist on parimate praktikate dokumendiga, mis on loodud pikajaaliste valikaadressi teenuse kasutajate poolt ja laiendab arusaamist sellest tehnoloogiast. Sisse toodi mõisted „kohalik

server“ ja „globaalne server“ ning „serverite skoop“. Kasutades BGP omadusi, on võimalik IP-prefikseid reklaamida nõnda, et need paistavad ainult teatud osale internetist, näiteks ainult ühes andmekeskuses, ühe konkreetse teenusepakkuja võrgus või teenusepakkujate kohtumiskohas (ing.k *Internet Exchange*). See annab juurde paindlikkust ja suurendab vastupidavust rünnakute vastu, kuna osade serveriteni ei saagi ründav liiklus kunagi jõuda ehk kliendid, keda tema teenindab, ei tunneta rünnet kogu süsteemi vastu, isegi kui kuskil mujal on teenus täielikult halvatud.

„On the Use of Anycast in DNS” [31]

Esimene omataoline töö, mis vastas küsimustele:

- Kas valikaadressiga päringu viide väheneb?
- Kas kliendid jõuavad lähima teenusserverini?
- Kas valikaadressi kasutajad omavad paremat käideldavust?

Lühidalt võib öelda, et nimepäringute viide vähenes, juhul kui kõik tsooni serverid olid globaalsed. Sellisel juhul jõuti peaaegu alati lähima serverini, aga sellega kaasnes ka oluliselt suurem raudvara ja tarkvara osakaal lahenduses, st sellega kasvasid teised riskid (keerukuse kasv). Siiski täheldati, et juhul kui mõni teenusserver oli kättesaamatu, siis pikenes tema „maas oleku” aeg, mida selgitab BGP suhteliselt pikk ajalõpp (ing.k *timeout*). Võib järeldada, et vaatluse all olevad serverid ei kasutanud BFD’d, et seda riski maandada.

„Stateful Anycast for DDoS Mitigation” [26]

Töö lahendab probleemi, et valikaadress ei sobi hästi TCP-põhiste teenuste kaitsmiseks. Töö autor pakub välja PIAS’i (ing.k *Proxy IP Anycast Service*), kus enne teenusservereid asuvad vaheserverid, mis saadavad liikluse determineeritult samade/õigete teenusserveriteni. See tagab TCP-sessioonide korrektse toimise ning lisaks sellele võimaldab vaheserverite kihis otsustada, kas liiklus on legitiimne või mitte ja tuleks hoopis tõkestada, mis aitab kaitsta teenusservereid. Väga sarnast tarkvaralist vaheserverite kihti kasutab tänapäeval ka Google oma koormusjaotuskihis [25].

2.6.2 Konfiguratsiooni haldus vahendite moodulid

Puppet-quagga [32] – Puppeti [33] (konfiguratsiooni haldusvahend) Quagga [34](BGP ruuteri tarkvara) moodul, millega saab seadistada Quaggat ja seeläbi luua BGP-ühendusi ning toetab ka Nagiose[35] monitooringut loodud BGP-sessioonidele.

Kuna ta on toodetud ICANN-i[36] poolt, kes on lähedalt seotud ka juurserverite haldusega, võib eeldada, et antud moodul on ka produktsioonis hästi testitud ja töökindel.

Puuduseks võib lugeda seda, et antud moodul haldab ainult teenusserveri poolset osa valikaadressi teenusest, lisaks on Quagga kasutus tänapäeval pigem vähenev ja tema baasil on loodud tänapäevasemaid lahendusi.

Lisaks on tegemist Puppeti mooduliga, st see eeldab, et kasutusele võetakse Puppet kogu infrastruktuuri halduseks. Puppet on küll üsna populaarne, aga tänapäeval on populaarsemaid vahendeid, milleni jõuame järgmises lõigus.

Ansible-role-exabgp [37] ja Ansible-exabgp [38] on mõlemad Ansible (konfiguratsiooni haldusvahend) rollid, millega on võimalik seadistada teenusserveris ExaBGP deemonit[39]. Mõlemad rollid omavad samalaadset funktsionaalsust, kuna ExaBGP on juba algselt mõeldud taolise lahenduse loomiseks (programmeeritavate BGP teenuste) ning lisaks on tal sisseehitatud teenuse toimivuse kontrolli võimekus, olles seega tegelikult väga hea kanditaat autori lahenduses kasutamiseks. Peamiseks põhjuseks, miks mitte ExaBGP'd kasutada, on see, et ta on mõeldud ainult teenusserverites kasutamiseks.

Ta ei ole oma võimalustelt võrreldav täisväärse BGP-deemoniga nagu Quagga või BIRD, kes haldavad ka kohalikku ruutingutabelit ja saavad seeläbi ka ruuterid olla. Tema peamiseks eesmärgiks on olla kihiks rakenduse ja BGP vahel. Tema integreerimine tulevikus üheks võimalikuks teenusserveri pooleks BGP-deemoniks on siiski väga kaalumist väärt mõte.[40]

2.6.3 Konfiguratsiooni generaatorid (BIRD seadistajad)

BCG [41] on ruuteri konfiguratsiooni seadistuse generaator BIRD tarkvarale, millega saab genereerida BGP-ühenduste seadistuse.

Puuduseks on nii teenuse juhtimise kui seire puudumine.

anycast_healthchecker [42] on deemon, mis tegeleb teenuse kättesaadavuse kontrolliga, võimaldab defineerida mitmeid teenuseid ning haldab nimekirja töötavatest teenuse IP-

aadressidest ja liidestub BIRD tarkvaraga, kui too on vastavalt seadistatud, ning seeläbi võimaldab luua valikaadressi teenuseid.

Eraldiseisvana täiesti mõistlik lahendus, mida kasutada teenusserverits ruutingudeemoni seadistamiseks ning teenuste toimimise kontrolliks. Siiski on ilmselt lihtsam võtta seal kasutusel olev loogika ja see luua konfiguratsioonihaldusvahendis kui lisada lahendusse üks täiesti eraldiseisev tarkvaraline komponent.

2.6.4 Terraform

Üheks võimaluseks, kuidas hallata infrastruktuuri on kasutada Terraformi [43], mis on vabavaraline tarkvara, millega luua IaC (ing.k *Infrastructure as Code*) lahendusi.

Tema peamiseks miinuseks on see, et ta eeldab API (*Application Programming Interface*) liidestuse kasutamise võimalikkust oma infrastruktuuri teenusepakkujaga ehk kui on soov sellist lahendust kasutada oma riistvaral, on vajalik lisatarkvara, mis pakuks API võimekust, mille kaudu saaks Terraformi kasutada.

Kusjuures on võimalik mitmeid näiteid, kuidas valikaadressi teenust luua, aga kõigi nende puhul on puuduseks see, et eeldatakse mingi konkreetset pilvepakkuja kasutamist, st kasutada saab ainult konkreetseid teenusepakkujaid. [44] [45] [46] [47]

Siiski üheks potentsiaaleks kasutusjuhuks võiks tulevikus olla Ansible rolliga Terraformi juhtimine ja seeläbi lahenduse kasutusala laiendamine erinevatele pilvedele.

Käesoleva töö vaates ta siiski ei sobi, kuna alternatiivsed lahendused pakuvad väga palju suuremat valikut tarkvara/raudvara, mida nad juhtida oskavad.

3 Nõuded ja eesmärgid

Selles peatükis kirjeldame ära nõuded lahendusele ning tarkvarakomplektile.

3.1 Nõuded loodavale lahendusele

Tulenevalt tehnoloogilisest baasist ning toetudes varasematele lahendustele ning analüüsides varasemaid töid, töötati välja nõuded lahendusele:

- Lahendust peab saama kasutada olenemata sellest, kas teenus asub füüsilisel, virtuaalsel või konteinertehnoloogial põhineval serveril.
- Lahendus peab võimaldama luua kõrgkäideldava teenuse, seega peab olema võimalik luua lahendus nõnda, et iga komponenti saab lisada süsteemi liiasusega.
- Lahendus peab võimaldama kasutada servereid erinevates andmekeskustes.
- Lahendus peab võimaldama toetatud tarkvarakomponentide ringi laiendamist.
- Lahendus peab võimaldama võimalikult lihtsalt erinevate komponentide (nimeserver / muu teenus / ruuter) välja vahetamist ja lisamist.
- Lahendus peab võimaldama enda testimist ning juurutatud lahenduse valideerimist.
- Lahendus peab toetama ilma agendita seadmete/tarkvara seadistamist.

3.2 Nõuded valitavale tarkvarakomplektile

Tarkvarakomplekt peab võimaldama lahendusele seatud nõuete täitmist.

Probleemipüstitusest tuleneb, et lahendus peab võimaldama hallata/luua teenust füüsilistel, virtuaalsetel ning konteineritel põhinevates serverites, seega saab kasutada ainult lahendusi, mis suudavad hallata kõiki neid kolme ja selleks on konfiguratsioonihaldusvahendid.

Johtuvalt sellest algab töö konfiguratsioonihaldusvahendi valimisest, tulenevalt nõuetest peab ta toetama võimalikult paljusid võrguseadmeid ja omama mooduleid võimalikult paljude teenusserverite tarkvarade juhtimiseks, et tagada lahenduse võimalikult lihtne laiendamine.

Kuna eesmärgiks on luua tööriist, mida hakatakse kasutama produktsioonisüsteemides, peab teda olema võimalik testida.

Tagamaks valitud tarkvarakomplekti aktuaalsus, võetakse üheks valikukriteeriumiks ka tarkvara populaarsus.

3.2.1 Eesmärgid

- Tarkvarakomplekti leidmine ja seadistamine, millega lahendus realiseerida.
- Konfiguratsioonivahendi leidmine.
- Testimise lahenduse valimine.

3.2.2 Nõuded

- Valitud tarkvara peab ühenduma teenusserveritega, kasutades SSH-protokolli.
- Valitud lahendus peab olema testitav Linux platvormil.
- Valitud ruuteritarkvara peab oskama olla nii teenusserver kui ruuter.
- Valitud lahendust peab saama laiendada toetama füüsilisi ruutereid.

4 Arenduse iteratsioonid

Selles peatükis luuakse mudel ning lähtuvalt lahendusele seatud nõuetest hakatakse seda rakendama.

Vastavalt valitud meetodile luuakse kõigepealt arenduse mudel, mille järgi hakata lahendust realiseerima.

Analüüsides nõudeid, otsustati vastavalt meetodile jagada nõuete täitmine erinevateks iteratsioonideks, kus igaüks käsitleb ühte tarkvarakomponenti. Igale iteratsioonile püstitatakse omad eesmärgid ja nõuded, mille järgi lahenduse luuakse ning hiljem valideeritakse. Valideerimist käsitletakse eraldi peatükis, mitte konkreetse iteratsiooni juures.

4.1 Iteratsioonide mudel

Lähtudes valitud metoodikast, leitakse igale iteratsioonile eesmärgid, vajalikud nõuded ning valideerimise tingimused, mille alusel nad täidetuks lugeda.

Iteratsiooni BIRDS eesmärgiks on toetada vähemalt ühte ruuteritarkvara ja selle kaudu luua BGP-ühendused ruuteri ja teenusserverite vahel.

Iteratsiooni NIMED eesmärgiks nimeserveri tarkvara juhtimine ning selle sidumine ruuteri tarkvaraga.

Iteratsioonide analüüs peaks vastama järgmisele mudelile:

Eesmärgid:

Nõuded:

Kusjuures eraldi peatükis on kirjeldatud iteratsiooni valideerimine.

4.2 Iteratsioon BIRDS

Selle iteratsiooni eesmärgiks on operatsioonisüsteemi ning marsruutimise rakendustarkvara seadistamise toe loomine.

4.2.1 Eesmärgid

1. Minimaalse lahenduse loomine, kasutades kolme Linuxi serverit tarkvarga BIRD, ja nende vahel BGP-ühenduse loomine.
2. Operatsioonisüsteemi seadistamine valikaadressi kasutamiseks vajalikul määral.
3. Erinevas rollis BIRD-i instantside seadistamine.

4.2.2 Nõuded

- Lahendus peab eristama erinevas rollis komponente.
- Lahendus peab olema testitav Linuxi platvormil.
- Lahendus peab genereerima konfiguratsiooni vastavalt rollile.

4.3 Iteratsioon NIMED

Käesolevas osas määratletakse nimeserveri tarkvara juhtimisega seonduvad eesmärgid ja nõuded.

4.3.1 Eesmärgid

- Nimeserveri tarkvara seadistamine.
- Nimeserveri tarkvara sidumine ruuteritarkvaraga.
- Nimeserveri teenuse seire.

4.3.2 Nõuded

- Lahendus peab olema testitav Linux platvormil.

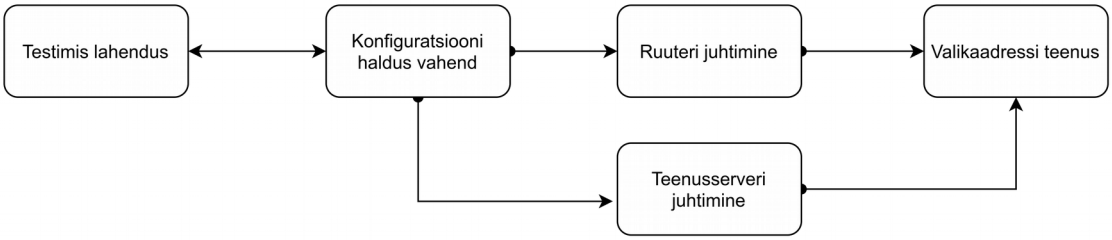
- Lahendus peab suutma signaliseerida ruuteri tarkvarale nimeserveri teenuse mitte töötamist.
- Valitud nimeserveri tarkvara peab toetama nii rekursiivset kui autoriteetset rolli.
- Nimeserveri tarkvara vastab serverid/hostname.bind päringule.

5 Realisatsioon

Käesolevas peatükis, toetudes nõuetele ning varasemate tööde analüüsile, valitakse/ luuakse/rakendatakse komponendid teenuse loomiseks.

5.1 Tarkvarakomplekti valimine

Siin leitakse tarkvaralised komponendid, mis sobivad esitatud eesmärkide ja nõuetega. Järgneval joonisel on näha ülevaatlik skeem lahenduse komponentidest.



Joonis 9.Lahenduse ülevaatlik skeem

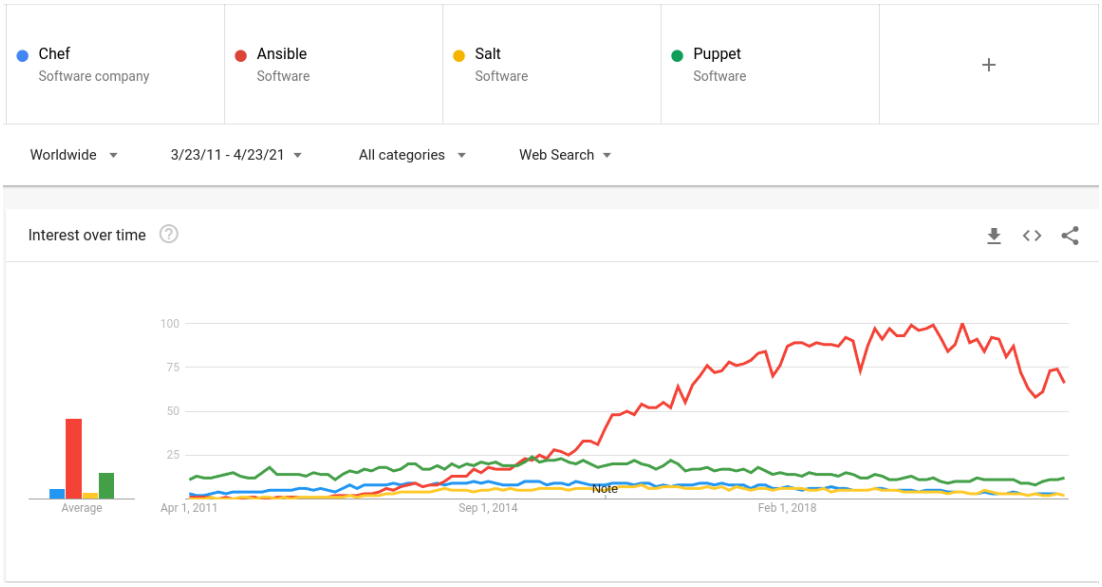
5.1.1 Konfiguratsioonihaldus vahend

Ansible [48] on üks levinumaid süsteemihaldusvahendeid. Tal on väga laialdane võrguseadmete tugi, on aktiivses arenduses ning kasutusel mitmes suures ettevõttes, kes seda projekti toetavad.

Peamiseks põhjuseks, miks valituks osutus Ansible, on tema väga suur tugi erinevatele võrguseadmetele, seda nii otse läbi enda moodulite kui ka läbi Napalm [49] raamistiku, isaks vastab ta põhilisele tingimusele, mis on üle SSH seadmete seadistamine. See tingimus välistas teise kõige lähema konkurendi Puppeti.

Lisaks eelnevale on Ansible'il loendamatu arv mooduleid, millega erinevaid tarkvarasid juhtida ja juurutada, mis muudab loodava lahenduse laiendamise oluliselt kiiremaks.

Nagu näha järgneval joonisel on viimase kümne aasta jooksul kõige populaarsem konfiguratsioonihaldusvahend olnud Ansible, millest võib järeldada, et ka tulevikus on tema kasutamine aktuaalne.



Joonis 10.Google Trends Ansible

Pole vähem oluline, et ka varasemates töödes on leitud tema sobivust võrguseadmete seadistamiseks. [50]

5.1.2 Testimis/valideerimise lahendus

Molecule [51] on Ansible'i jaoks loodud rollide testimise lahendus, mis võimaldab rolle testida väga erinevaid tehnoloogiaid kasutades. Tal on tugi Dockerile (konteinerite haldus vahend) ja Virtualboxile (Virtualiseerimise vahend), mis tähendab, et temaga on võimalik testida nii konteineritel kui ka virtuaalmasinatel.

Lisaks on testimiseks ja arenduseks kasutusel Vagrant. See on tööriist, millega saab luua ja hallata taasloodavaid arenduskeskkondi. [52] Peamiseks põhjuseks, miks Vagrant lahenduse komplekti sai kaasatud, on tema omadus, et ta suudab hallata nii konteinereid kui ka virtuaalmasinaid, mis tähendab, et tulevikus on võimalik lahendust testida ka platvormidel, mida ei saa konteinerites kasutada. [53]

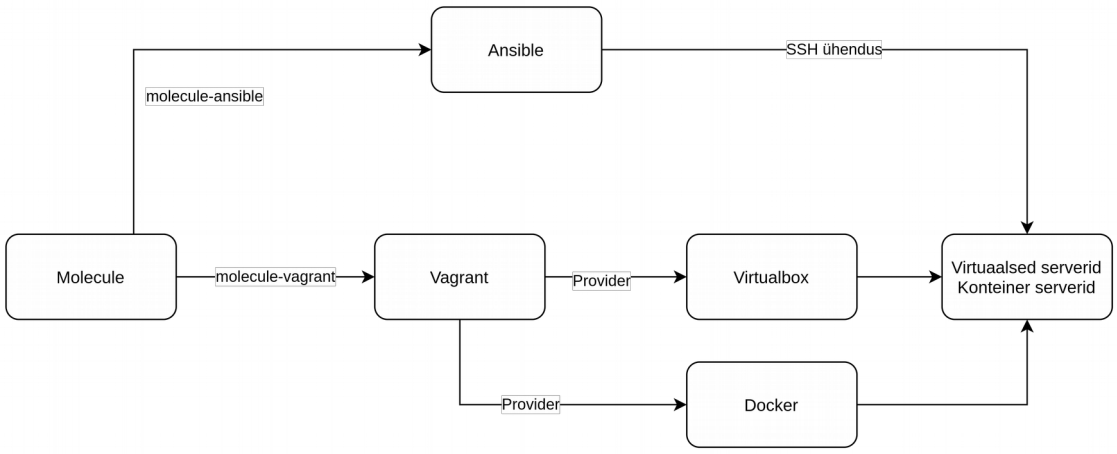
Peamiseks põhjuseks, miks testimiseks valiti Molecule, on tema platvormide tugi ning ülesehitus, mis võimaldab luua palju erinevaid olukordi (ing.k *scenario*), millel teste läbi viia,

mis tähendab seda, et saame valideerida igas iteratsioonis defineeritud osiseid eraldi ning lõpuks nad kokku siduda terviklikuks lahenduse testiks.

Testimise tarkvara täistestitsükli puhul läbitakse järgnevad sammud:

1. Kõigepealt lahendatakse sõltuvused, mis on defineeritud (laeb alla vajalikud rollid).
2. Kontrollitakse rollide süntaksit.
3. Kustutatakse kõik vanad testimise instantsid.
4. Luuakse uued instantsid.
5. Käivitatakse „prerun" skript, millega saab sisse viia vajalikud testieelsed muudatused.
6. Rakendatakse testitavat rolli (converge).
7. Rakendatakse rolli teist korda, et veenduda, et tulemus ei muutu (idempotence).
8. Käivitatakse „side_effect" skript, millega saab imiteerida mingeid konkreetseid veaolukordi.
9. Valideeritakse tulemust.
10. Kustutatakse loodud instantsid.

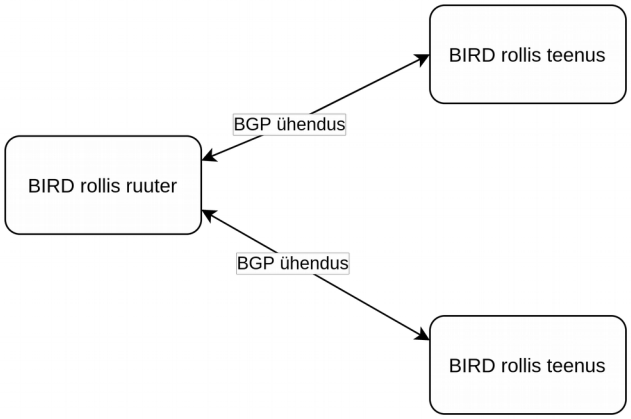
Järgneval joonisel on ülevaatlik skeem kogu testimislahendusest.



Joonis 11.Testimise lahenduse skeem

5.2 Iteratsioon BIRDS

Käesolevas osas realiseeritamse tarkvaraline lahendus, mis seadistab tarkvaralised ruuterid ja loob nende vahele BGP-ühenduse ning valideerib selle toimimist. Joonisel on näha ülevaade valideerimiseks kasutatud osistest.



Joonis 12.BIRDS skeem

5.2.1 Rakendamine

Esimeseks suuremaks küsimuseks osa realiseerimisel oli kas lahendada ruuteri juhtimine kirjutatava rolli piires või integreerida mõni varasem roll siia lahendusse. Peale mõlema variandi proovimist otsustati siiski minna varem kirjutatud rolli kasutamise teed, kuna nõnda on võimalik saavutada suurem funktsionaalsuse toetamine väiksema kuluga ehk peamiseks keerukuseks oli lahenduse nõude täitmine, et toetatud peavad olema nii teenusserveri kui ruuteri rollis BIRD-instantsid.

Selle saavutamiseks käivitatakse „ansible-bird“[54] rolli korduvalt, olenevalt kas server, millele rolli rakendatakse, on „service_role“ tüübiga „routers“ või „services“. „Routers“ tüübi puhul luuakse talle BGP-seadistus, milles on kõik „services“ rolli kuuluvad serverid. Kui tüübiks on „services“, luuakse talle BGP-seadistus, kus on kõik „routers“ rolli kuuluvad serverid ning lisatakse talle „service_ip’ga“ defineeritud teenuse IP. Valideerimist käsitletakse eraldi peatükis.

5.2.2 Lisandunud omadused/funktsionaalsus

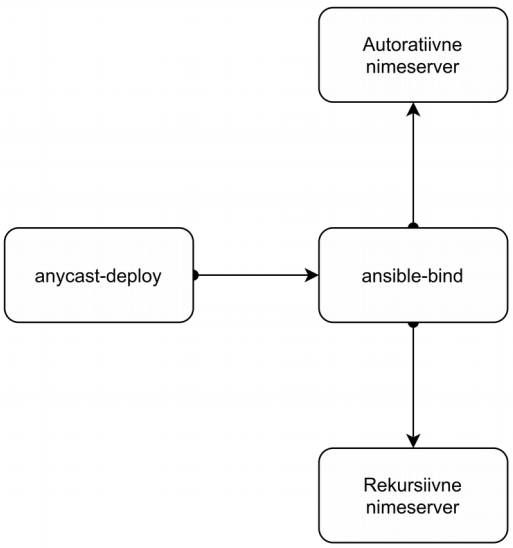
service_role atribuut routers / services mille abil eristada erinevas rollis olevaid komponente.

router_id tuleneb esimesel võrguliidesel olevast IP aadressist.

service_ip valikaadressi teenuse IP, millelt teenus on kättesaadav mis teenusserveris ning mille vastu teaostatakse kättesaadavuse kontroll peale BGP ühenduse loomist ruuterist.

5.3 Iteratsioon NIMED

Selle iteratsiooni eesmärgiks oli luua nimeserveri tarkvara juhtiv funktsionaalsus. Järgneval joonisel on skeem loodud lahendusest:



Joonis 13.Nimeserveri juhtimise skeem

„anycast-deploy"[55] roll kasutab „ansible-bind"[56] rolli, et seadistada nimeserver vastavalt „nameserver_role" atribuudile kas autoriteetseks või rekursiivseks.

5.3.1 Rakendamine

Esimeseks küsimuseks nagu ka eelmises iteratsioonis oli kas luua oma nimeserverit juhtiv funktsionaalsus või taaskasutada mõnda olemas olevat rolli, otsustatud sai kasutada olemas olevat rolli „ansible-bind". Sarnaselt eelmisele iteratsioonile käivitatakse „ansible-bind" rolli vastavalt „nameserver_role" atribuudile parameetritega, millega ühel juhul seadistatakse autoriteetne nimeserver, teisel juhul rekursiivne.

Serverisse paigaldatakse skript mis luuakse vastavalt seadistatud „anycast_testing_record", „anycast_testing_record_response" ja „service_ip" muutujates olevatele andmetele, millega kontrollitakse nimeserveri vastamist ja ebaõnnestunud vastuse puhul peatatakse ruutingu deemoni töö, seda skripti käivitatakse vastavalt „healthcheck_interval" parameetrile.

Valideerimislahendust arendades ilmnes, et Ansible'i sisseehitatud nimelahendamise moodul lahendab nimesid serveris, kus Ansible'it käivitatakse, seega ei sobi ta antud lahendusse, kuna soov on valideerimise käigus lahendada nimesid just nimelt seal serveris, kuhu parasjagu rolli rakendatakse.

Seega tuli kasutusele võtta kogukonna poolt loodud teine dig moodul (alexisfacques.ansible_module_dig), mis võimaldas vajalikul viisil nimesid lahendada. Leitud

lahendus on hea näide Ansible'i modulaarsusest - isegi kui mõni tema põhikomponent ei käitu kasutajale vajalikult, on võimalik leida alternatiivne lahendus.

5.3.2 Lisandunud omadused/funktsionaalsus

„anycast_testing_record": ns1.testing.local (päring, mis esitatakse teenusserverile)

„anycast_testing_record_response": 127.0.0.1 (oodatud vastus päringule, mis saadeti teenusserverile)

„healthcheck_interval" : kui tihti käivitatakse nimeserveri elusoleku kontrolli.

„hostname.bind" võrdlemine teenusserveri nimega. (vajas „ansible-bind" rolli laiendamist)

Nimeserver tööd kontrolliva ning ruuteri tarkvara juhtiva skripti loomine.

6 Valideerimine ja tulemused

Selles peatükis käsitletakse iteratsioonide valideerimist, mis põhineb analüüsi käigus tekkinud nõuetele. Vaadeldakse tulemusi ning antakse hinnang realisatsioonile.

Tulenevalt analüüsi käigus tekkinud nõuetest luuakse valideerimise tingimused ning realiseeritakse nad loodud tööriistas funktsionaalsusena. Osa valideerimise funktsionaalsusest on kaetud juba tarkvara seadistavas osas ja seda ei duplitseeritud arendatud rollis.

Iga iteratsiooni kohta on oma alampeatükk.

6.1 Iteratsiooni BIRDS valideerimine

Ruuteri ning operatsioonisüsteemi seadistamise valideerimise tingimused tulenesid nõuetest/eesmärkidest antud iteratsioonis.

Lahenduse valideerimiseks oli vaja kolme ruuterit, millest üks oli rollis ruuter ja kaks rollis teenusserver.

6.1.1 Valideerimise tingimused

Valideerimise tingimused on järgmised:

1. Ruuteri tarkvara töötab.
2. Ruuteri ja teenuste vahel töötab icmp ping.
3. Teenusserverites on seadistatud teenuse IP-aadress.
4. Ruuteri ja teenusserverite vahel on BGP-ühendus.
5. Ruuter saab pingida teenuse IP’d.

6.1.2 Realisatsioon

Ruuteri tarkvara töötamine valideeritakse juba ruuteri seadistamise käigus „ansible-bird" rolli poolt.

Kõik järgnev valideerimine realiseeriti „anycast-deploy" rollis.

Ruuteri ja teenusserverite vaheline icmp pingi kontroll teostati routers rollis olevast serverist, pingides kõiki services rollis olevaid servereid.

Teenusserveris kontrollitakse, kas silmusseadmele on määratud teenuse IP-aadress ja seejärel testitakse seda, pingides teda samast teenusserverist.

BGP-ühendust ruuteri ja teenusserverite vahel kontrollitakse teenusserveris, vaadeldes BIRD tarkvara aktiivseid BGP-seansse.

Viimase sammuna kontrollitakse, kas ruuter saab pingida teenuse IP-aadressi.

6.2 Iteratsiooni NIMED valideerimine

Nagu eelpool mainitud, ilmnesid selle valideerimise arendamisel Ansible'i iseärasused, aga siiski õnnestus kõik kontrollid realiseerida.

6.2.1 Valideerimise tingimused

1. Nimeserveri tarkvara töötab.
2. Nimeserver vastab nõutud päringule oodatult.
3. Nimeserver vastab hostname.bind päringule oma nimega.
4. Nimeserveri tõrke puhul ruuteri tarkvara lõpetab teenuse reklaamimise antud serverist.

6.2.2 Realisatsioon

Nimeserveri tarkvara töötamise kontroll realiseeritakse juba „ansible-bind" rollis.

Nimeserveri vastamine nõutud päringule oodatult realiseeriti „anycast-deploy" rollis ja seda läbi seadistatavate parameetrite „anycast_testing_record" ning „anycast_testing_record_response",

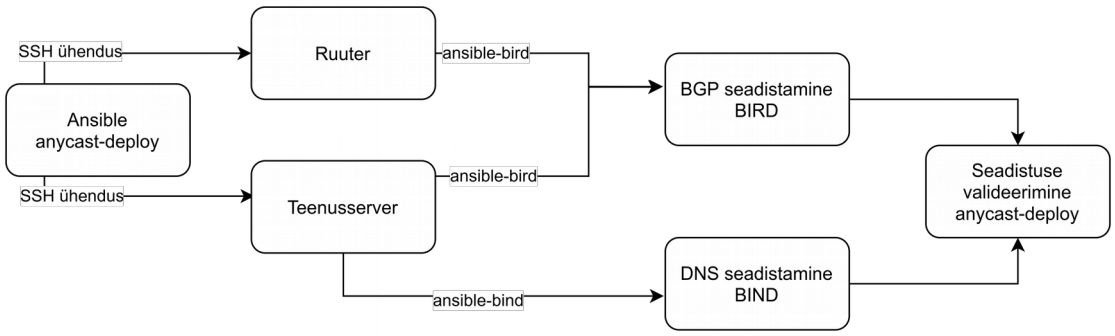
teenusserveris päritakse „service_ip” aadressilt „anycast_testing_recordit” ja vastuseks oodatakse tulemust, mis on kirjas „anycast_testing_record_response” parameetris.

Nimeserveri vastamist päringule hostname.bind valideeritakse ruuterist, pärides kõigilt teenusserveritelt nende haldus IP aadressi kasutades hostname.bind päring ja vastuseks oodatakse sama nime mis on teenusserveril.

Nimeserveri tõrke puhul ruuteri tarkvara peatamist valideeritakse läbi Molecule „side_effect” funktsionaalsuse, millega peatatakse nimeserver ja seejärel käivitatakse nimeserveri elusoleku kontrollskript, mille peale ruuteri tarkvara peatatakse.

6.3 Tulemused

Tulemuste hindamiseks peame kõigepealt vaatlema realiseeritud lahenduse komponentide skeemi, mis asub järgneval joonisel.



Joonis 14.Realiseeritud lahenduse skeem

Konfiguratsioonihaldusvahendi valikut võib lugeda edukaks, kuna täitis kõiki seatud nõudmisi, millest peamised olid, et ühendus seadistatavasse serveritesse kasutab SSH protokolli ning ei vaja toimimiseks agente. Lahenduse realiseerimine Ansible’i rollina „anycast-deploy” oli kindlasti õige valik, seda kinnitasid isegi ilmnenud probleemid, sest neist kõiki oli võimalik suhteliselt kerge vaevaga ületada just tänu valitud vahendi paindlikkusele.

Edukalt realiseeriti ruuteritarkvara juhtimine, seda nii rollis ruuter kui ka rollis teenusserver, kusjuures õnnestus taaskasutada ruuteri seadistamiseks mõeldud rolli „ansible-bird”. Kahjuks küll ei õnnestunud jõuda mitme ruuteritarkvara toetamiseni, aga arvestades töö mahtu oli see oodatav tulemus.

Samuti realiseeriti edukalt nimeserveri teenuse juhtimine rolliga „ansible-bind”, mille puhul pidi küll olemas olevat rolli funktsionaalsust mõnevõrra laiendama, aga ikkagi oli selle töö maht

oluliselt väiksem, kui oleks olnud nullist sellise funktsionaalsuse kirjutamine. Võimalik on nimeserverit seadistada nii rekursiivseks kui autoriteetseks.

Valideerimiseks Molecule'i kasutamine õigustas ennast täielikult, kuna võimaldas luua iteratsioonispetsiifilisi keskkondi, millel lahendust valideerida, kasutades „anycast-deploy" rollis loodud funktsionaalsust, mida kasutatakse nii lahenduse rakendamisel kui ka testide läbiviimisel.

7 Lahenduse laiendamine

Mõistlikke laiendusi loodud lahendusele on muidugi rohkem, kui siinkohal autor jõuab üles märkida, aga proovitakse nad kasulikkuse järgi reastada ja natukene nende sisu avada.

Peamiseks puudulikkuseks peab autor hetkel seda, et ükski raudvaraline ruuter ei ole toetatud. See ei takista küll lahenduse kasutuselevõttu, aga võiks siiski olla lahenduse laiendamisel järgmine samm. Kõige asjakohasem oleks Junose toetamine, kuna seda on võimalik testida, kasutades Junose Vagranti masinat [57].

Kindlasti oleks hea valik ka testimise/valideerimise lahenduse laiendamine „molecule-goss"'i [58] kasutades. See aitaks standardiseerida kasutusel olevate rakendustarkvarade seadistuse valideerimist, seda, kas nad kuulavad õigetel võrguliidestel ja kas deemonid on töötavas olekus.

Vähemtähtsaks ei saa pidada erinevate nimeserveri tarkvarade toe laiendamist, esikohal oleks siin kindlasti PowerDNS, kuna sel on tootjapoolsed Ansible'i rollid rakendustarkvara rakendamiseks ja seadistamiseks.

Kaugemas perspektiivis oleks üheks võimalikuks teeks hallata ka tsoonifaili, mida lahendusega paigaldatakse. Siinkohal oleks igati asjakohane realiseerida ka DNSSEC-i tugi ja tulemit valideerida, integreerides Zonemaster.

Kasulik oleks ka kaaluda Ansible'i kasutamist, et juhtida Terraformi ja seeläbi tekitada võimekus paigutada teenus erinevate pilvepakkujate infrastruktuuridele.

Üheks potentsiaalselt huvitavaks tuleviku uurimussuunaks oleks BGP kaudu Kubernetese klastrites paiknevate teenusserverite toetamine lahenduses ning seejärel nende lahenduste jõudluse võrdlus. [59]

8 Kokkuvõte

Peamiseks eesmärgiks oli luua tööriist, millega on võimalik taaskorratavalt luua valikaadressi põhist nimeteenust ning valideerida selle toimivust simulatsioonikeskkonnas, mis ka edukalt õnnestus.

Selleni jõudmiseks oli oluline tutvuda varasemate teoreetiliste töödega, mis kajastasid seotud tehnoloogiaid ja valikaadressi ennast. Lisaks tutvus autor sarnast või osalist funktsionaalsust pakkuvate alternatiivsete lahendustega, ja kaalus nende kasutamist oma lahenduse loomisel. Peamiselt analüüsides varasemaid töid ning, toetudes praktilisele kogemusele loodi nõuded kogu lahendusele.

Kasutades valitud analüütilist metoodikat, loodi mudel, mille järgi arendust juhtida ja tulemusi valideerida. Seejärel realiseeriti analüüsist tekkinud nõuded ja eesmärgid arendusiteratsioonidena, mille tulemuseks on Ansible'i roll „anycast-deploy" ehk tööriist, millega sellist lahendust juurutada.

Valminud lahendus kasutab ära eelnevalt kogukonna poolt loodud Ansible'i rolle „ansible-bird" ja „ansible-bind" millest viimane nõudis ka autori poolset laiendamist, et sobituda lahendusse. Kolmanda osapoolte rollide kasutamine näitab kuidas tulevikus lahenduse funktsionaalsust laiendada, st on märk tema modulaarsusest.

Tulemuseks olev tarkvarakomplekt võimaldab enda testimist ja lahenduse toimimise valideerimist, kusjuures valideerimise funktsionaalsus on lahendatud peaasjalikult just „anycast-deploy" rolli raames, ehk siis on näiteks kuidas erinevate osade valideerimist saab arendada nõnda, et nad ei sõltu konkreetsetest komponentidest mis hetkel kasutusel on.

Taaskorratava valikaadressipõhise nimeteenuse juurutamise lahenduse loomiseks vajalike nõuete ja eesmärkide määratlemine ning realiseerimine oli edukas.

Ainukeseks puuduseks võib lugeda toetatud komponentide vähesust, aga siinkohal sai piiravaks teguriks töö maht.

Kasutatud kirjandus

[1] „root-servers.org“, *Root-servers*. https://root-servers.org/ (vaadatud märts 28, 2021).
[2] Internet Systems Consortium, Inc., „A Software Approach to Distributing Requests for DNS Service using GNU Zebra, ISC BIND 9 and FreeBSD“, 2004, [võrgus]. Kättesaadav: https://www.isc.org/pubs/tn/isc-tn-2004-1.html#appx.Routers
[3] Alan R. Hevner, Salvatore T. March, Jinsoo Park and Sudha Ram, „Design Science in Information Systems Research“, *MIS Q.*, kd 28, nr 1, lk 75–105, 2004, doi: https://doi.org/10.2307/25148625.
[4] T. W. Edgar ja D. O. Manz, *Research methods for cyber security*. Cambridge, MA: Syngress, an imprint of Elsevier, 2017.
[5] https://root-servers.org/, „A Highly-Available Multiple Region Multiaccess Edge Computing Platform with Traffic Failover“, Master's Thesis, Aalto University School of Science, Espoo, 2020. Vaadatud: veebr 01, 2021. [võrgus]. Kättesaadav: https://aaltodoc.aalto.fi/bitstream/handle/123456789/46124/master_Sulaeman_Adika_2020.pdf?sequence=1&isAllowed=y
[6] J. Postel, „Internet Protocol“, RFC Editor, RFC0791, sept 1981. doi: 10.17487/rfc0791.
[7] J. Postel, „Transmission Control Protocol“, RFC Editor, RFC0793, sept 1981. doi: 10.17487/rfc0793.
[8] J. Postel, „User Datagram Protocol“, RFC Editor, RFC0768, aug 1980. doi: 10.17487/rfc0768.
[9] J. C. Mogul, „Broadcasting Internet Datagrams“, RFC Editor, RFC0919, okt 1984. doi: 10.17487/rfc0919.
[10] P. Savola, „Overview of the Internet Multicast Routing Architecture“, RFC Editor, RFC5110, jaan 2008. doi: 10.17487/rfc5110.
[11] A. Begen ja T. Stockhammer, „Guidelines for Implementing Digital Video Broadcasting - IPTV (DVB-IPTV) Application-Layer Hybrid Forward Error Correction (FEC) Protection“, RFC Editor, RFC6683, aug 2012. doi: 10.17487/rfc6683.
[12] C. Partridge, T. Mendez, ja W. Milliken, „Host Anycasting Service“, RFC Editor, RFC1546, nov 1993. doi: 10.17487/rfc1546.
[13] „Akit: IP-Prefiks“, mai 15, 2021. https://akit.cyber.ee/term/14228-ip-prefix
[14] P. V. Mockapetris, „Domain names - concepts and facilities“, RFC Editor, RFC1034, nov 1987. doi: 10.17487/rfc1034.
[15] P. V. Mockapetris, „Domain names - implementation and specification“, RFC Editor, RFC1035, nov 1987. doi: 10.17487/rfc1035.
[16] „StatsDNS“, apr 28, 2021. https://www.statdns.com/rfc/ (vaadatud apr 28, 2021).
[17] K. Lougheed ja Y. Rekhter, „Border Gateway Protocol (BGP)“, RFC Editor, RFC1105, juuni 1989. doi: 10.17487/rfc1105.
[18] „RIPE NCC“. https://www.ripe.net/ (vaadatud apr 28, 2021).
[19] „AS Assignment“. https://en.wikipedia.org/wiki/Autonomous_system_(Internet)#Assignment
[20] J. Mitchell, „Autonomous System (AS) Reservation for Private Use“, RFC Editor, RFC6996, juuli 2013. doi: 10.17487/rfc6996.
[21] T. Bates, E. Chen, ja R. Chandra, „BGP Route Reflection: An Alternative to Full Mesh Internal BGP (IBGP)“, RFC Editor, RFC4456, apr 2006. doi: 10.17487/rfc4456.
[22] D. Katz ja D. Ward, „Bidirectional Forwarding Detection (BFD) for IPv4 and IPv6 (Single Hop)“, RFC Editor, RFC5881, juuni 2010. doi: 10.17487/rfc5881.
[23] C. Hopps, „Analysis of an Equal-Cost Multi-Path Algorithm“, RFC Editor, RFC2992, nov 2000. doi: 10.17487/rfc2992.
[24] *RFC1546*. [võrgus]. Kättesaadav: https://tools.ietf.org/html/rfc1546
[25] D. E. Eisenbud *et al.*, „Maglev: A Fast and Reliable Software Network Load Balancer“, *13th USENIX Symposium on Networked Systems Design and Implementation (NSDI 16)*, Santa Clara, CA, märts 2016, lk 523–535. [võrgus]. Kättesaadav: https://www.usenix.org/conference/nsdi16/technical-sessions/presentation/eisenbud
[26] R. Hansen, „Stateful Anycast for DDoS Mitigation“, 2007.
[27] Trohar, Tomislav and Silvio Papić, „Alternative to using VRRP for Mutual Next-Hop Redundancy“, *Int. J. Digit. Technol. Econ.*, kd 2, nr 2, lk 123–126, 2017.

[28] „uCARP“. https://github.com/jedisct1/UCarp (vaadatud mai 05, 2021).
[29] J. Abley ja K. Lindqvist, „Operation of Anycast Services“, RFC Editor, RFC4786, dets 2006. doi: 10.17487/rfc4786.
[30] T. Hardie, „Distributing Authoritative Name Servers via Shared Unicast Addresses“, RFC Editor, RFC3258, apr 2002. doi: 10.17487/rfc3258.
[31] S. Sarat, V. Pappas, ja A. Terzis, „On the use of anycast in DNS“, *ACM SIGMETRICS Perform. Eval. Rev.*, kd 33, nr 1, lk 394–395, juuni 2005, doi: 10.1145/1071690.1064271.
[32] ICANN, „puppet-quagga“. https://forge.puppet.com/modules/icann/quagga (vaadatud apr 29, 2021).
[33] „Puppet“. https://puppet.com/ (vaadatud juuni 10, 2020).
[34] „Quagga“. https://www.quagga.net (vaadatud apr 29, 2021).
[35] „Nagios“. https://www.nagios.org/ (vaadatud apr 29, 2021).
[36] „ICANN“. https://www.icann.org/ (vaadatud apr 29, 2021).
[37] *Ansible-role-exabgp*. Vaadatud: apr 29, 2021. [võrgus]. Kättesaadav: https://github.com/hybridadmin/ansible-role-exabgp
[38] *Ansible-exabgp*. Vaadatud: apr 29, 2021. [võrgus]. Kättesaadav: https://github.com/elcomtik/ansible-exabgp
[39] *ExaBGP*. Vaadatud: apr 03, 2021. [võrgus]. Kättesaadav: https://github.com/Exa-Networks/exabgp
[40] Anton Aksola, „ExaBGP Introduction and Real Life Use Cases“, 2016. https://www.trex.fi/2016/exa-bgp-trex.pdf (vaadatud apr 03, 2021).
[41] *Bird Config Generator*. [võrgus]. Kättesaadav: https://github.com/natesales/bcg
[42] *Anycast_Healthchecker*. Vaadatud: apr 29, 2021. [võrgus]. Kättesaadav: https://github.com/unixsurfer/anycast_healthchecker
[43] „Terraform Tarkvara“. https://en.wikipedia.org/wiki/Terraform_(software) (vaadatud apr 29, 2021).
[44] „Terraform AdvanceHosting“. https://registry.terraform.io/providers/advancedhosting/ah/latest/docs/resources/ah_ip (vaadatud apr 29, 2021).
[45] „Terraform Packet“. https://github.com/atoonk/packet_anycast (vaadatud apr 29, 2021).
[46] „Terraform AWS“. https://faun.pub/building-a-high-available-anycast-service-using-aws-global-accelerator-450fc8c4fd1e (vaadatud apr 29, 2021).
[47] „Terraform GCP“. https://ahmet.im/blog/cloud-run-multi-region-terraform/ (vaadatud apr 29, 2021).
[48] „Ansible“. https://www.ansible.com/ (vaadatud juuni 10, 2020).
[49] *Napalm Network Automation*. Vaadatud: apr 29, 2021. [võrgus]. Kättesaadav: https://napalm-automation.net/
[50] Mikk Aruoja, „Võrguseadmete konfigureerimise automatiseerimine“, Tallinna Tehnikaülikool, Tallinn, 2020. Vaadatud: apr 01, 2020. [võrgus]. Kättesaadav: https://digikogu.taltech.ee/et/Item/7e3d98e3-20ac-4f75-887b-d562eb9f7429
[51] *Molecule*. Vaadatud: apr 29, 2021. [võrgus]. Kättesaadav: https://molecule.readthedocs.io/en/latest/configuration.html
[52] HashiCorp, *Vagrant*. Vaadatud: apr 29, 2021. [võrgus]. Kättesaadav: https://www.vagrantup.com/intro
[53] „Molecule Testing“. https://sbarnea.com/slides/molecule/#/14 (vaadatud apr 29, 2021).
[54] Logan V, *ansible-bird*. Vaadatud: veebr 03, 2021. [võrgus]. Kättesaadav: https://github.com/teadur/ansible-bird
[55] Georg Kahest, *anycast-deploy*. Vaadatud: mai 01, 2021. [võrgus]. Kättesaadav: https://github.com/teadur/anycast-deploy
[56] René Moser, *ansible-bind*. Vaadatud: jaan 01, 2021. [võrgus]. Kättesaadav: https://github.com/teadur/ansible-role-bind
[57] „Vagrant QFX“. https://app.vagrantup.com/juniper/boxes/vqfx10k-re
[58] *Molecule Goss*. Vaadatud: apr 29, 2021. [võrgus]. Kättesaadav: https://github.com/ansible-community/molecule-goss
[59] Johani Vajakas, „Kubernetese võrgukihi tehnoloogiate jõudluse võrdlemine füüsilistel serveritel“, TARTU ÜLIKOOL, Tartu, 2020. [võrgus]. Kättesaadav: https://www.ims.ut.ee/www-public2/at/2020/bsc/atprog-courses-bakalaureuset55-loti.05.029-johani-vajakas-text-20200519.pdf
[60] „Wikipedia Routing Delivery schemes“. https://en.wikipedia.org/wiki/Routing#Delivery_schemes (vaadatud apr 29, 2021).

Lisa 1– Lihtlitsents lõputöö reprodutseerimiseks ja lõputöö üldsusele kättesaadavaks tegemiseks[1]

Mina, Georg Kahest

1 Annan Tallinna Tehnikaülikoolile tasuta loa (lihtlitsentsi) enda loodud teose „Korratav valikaddressipõhine nimelahenduse teenus" mille juhendaja on Toomas Lepik

1.1 reprodutseerimiseks lõputöö säilitamise ja elektroonse avaldamise eesmärgil, sh Tallinna Tehnikaülikooli raamatukogu digikogusse lisamise eesmärgil kuni autoriõiguse kehtivuse tähtaja lõppemiseni;

1.2 üldsusele kättesaadavaks tegemiseks Tallinna Tehnikaülikooli veebikeskkonna kaudu, sealhulgas Tallinna Tehnikaülikooli raamatukogu digikogu kaudu kuni autoriõiguse kehtivuse tähtaja lõppemiseni.

2 Olen teadlik, et käesoleva lihtlitsentsi punktis 1 nimetatud õigused jäävad alles ka autorile.

3 Kinnitan, et lihtlitsentsi andmisega ei rikuta teiste isikute intellektuaalomandi ega isikuandmete kaitse seadusest ning muudest õigusaktidest tulenevaid õigusi.

17.05.2021

1 Lihtlitsents ei kehti juurdepääsupiirangu kehtivuse ajal vastavalt üliõpilase taotlusele lõputööle juurdepääsupiirangu kehtestamiseks, mis on allkirjastatud teaduskonna dekaani poolt, välja arvatud ülikooli õigus lõputööd reprodutseerida üksnes säilitamise eesmärgil. Kui lõputöö on loonud kaks või enam isikut oma ühise loomingulise tegevusega ning lõputöö kaas- või ühisautor(id) ei ole andnud lõputööd kaitsvale üliõpilasele kindlaksmääratud tähtajaks nõusolekut lõputöö reprodutseerimiseks ja avalikustamiseks vastavalt lihtlitsentsi punktidele 1.1. ja 1.2, siis lihtlitsents nimetatud tähtaja jooksul ei kehti.