TALLINNA TEHNIKAÜLIKOOL
Infotehnoloogia teaduskond

Gerd Kukemilk 091295IASB

Voogtelemeetrial põhinev võrguseadmete seire Tervise ja Heaolu Infosüsteemi Keskuse näitel

Bakalaureusetöö

Juhendaja: Priit Raspel
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: Gerd Kukemilk

03.01.2021

Annotatsioon

Töö eesmärgiks on uurida voogtelemeetria erinevaid protokolle ja leida nendest parim, seejärel rakendada valitud voogtelemeetria protokoll TEHIK riigiasutuses võrguseire süsteemides.

Selle eesmärgi saavutamiseks kajastatakse esmalt teemakohast kirjandust, mis sisaldab voogtelemeetria protokollide ülevaadet, seejärel asutuses kasutusel olevate seirerakenduste tausta, mille põhjalt on edasises töös võimalik analüüs koostada.

Analüüsi käigus lahatakse TEHIKu võrguseire olukorda, võrreldakse NETCONF-i ja gNMI-d, ning järeldatakse, et viimane voogtelemeetria protokoll on käsitletava asutuse võrguseire teostamiseks sobivaim.

Viimastes osades kavandatakse ja realiseeritakse, labori keskkonnas, voogtelemeetria põhine seire süsteem, mis peegeldab TEHIKu reaalseid võrguseire vajadusi ja keskkonda.

Lõputöö on kirjutatud eesti keeles ning sisaldab teksti 49 leheküljel, 5 peatükki, 31 joonist, 6 tabelit.

Abstract

Streaming Telemetry Based Monitoring of Network Devices on the Example of Health and Welfare Information Systems Centre

The goal of this paper is to examine different protocols that can be used to send Streaming Telemetry from network devices, then apply the chosen protocol for the purpose of monitoring the network of Health and Welfare Information Systems Centre.

The first part of this paper reviews various literature on network monitooring and Streaming Telemetry in order to create a basis for the later analysis of these topics.

The analysis examines the current situation of network monitooring and the problems that currently exist. Having done that it compares NETCONF and gNMI Stream Telemetry capabilities.

The last parts of this paper focus on the planning of a Streaming Telemetry based monitoring solution and implementation in a test environment.

The thesis is written in Estonian and contains 49 pages of text, 5 chapters, 31 figures, 6 tables.

Lühendite loend

CPU	*Central Processing Unit*, Protsessor
DoD	*Department of Defence,* Ameerika Ühendiriikide kaitseministeerium
gNMI	*gRPC Network Management Interface,* gRPC võrguhaldusliides
gRPC	*gRPC Remote Procedure Call,* gRPC kaugprotseduurikutse
IAB	*Internet Activities Board*, Interneti arhitektuuri nõukogu
IP	*Internet Protocol,* Interneti protokoll
ISO	*International Organization for Standardization,* Rahvusvahelisele Standardimisorganisatsioonile
IT	Infotehnoloogia
JSON	*Javascript Object Notation*, Javascript objektide notatsioon
MIB	*Management Information Base*, haldusinformatsiooni baas
NETCONF	*Network Configuration*, võrgukonfiguratsioon
OID	*Object identifier hierarchy*, objekti identifikaatorite hierarhia
PromQL	*Prometheus Query Language*, Prometheuse päringukeel
RFC	*Request For Comments*
RPC	*Remote Procedure Call*, kaugprotseduurikutse
SGMP	*Simple Gateway Monitoring Protocol,* lihtne lüüsiseire protokoll
SMI	*The Structure of Management Information,* haldusinfo struktuur
SNMP	*Simple Network Management Protocol*, lihtne võrguhaldus protokoll
SQL	*Structured Query Language,* Struktureeritud päringukeel
TEHIK	Tervise ja heaolu infosüsteemide keskus
UDP	*User Datagram Protocol*, Kasutajadatagrammi protokoll
XML	*Extensible Markup Language,* laiendatav märgistuskeel
YAML	*"*YAML *Ain't Markup Language"*
YANG	*Yet Another Next Generation,* Veel üks uus generatsioon

Sisukord

Jooniste loetelu

Tabelite loetelu

Sissejuhatus

Selle töö kirjutamise ajendiks sai asjaolu, et Tervise ja Heaolu Infosüsteemide Keskuses (edaspidi TEHIK) on vajadus luua parem, kiirem, täpsem ja väiksema ressursikuluga võrguseadmete seiresüsteem.

Käesoleval ajal on võrguseire osatähtsus üle maailma erinevates asutustes olulise tähelepanu all, seda eelkõige aegunud tehnoloogiate kasutamise tõttu nagu *Simple Network Management Protocol* (edaspidi SNMP). Arvestades, et asutustes ja organisatsioonides pole mitte ainult kasvav seadmete kogus, vaid ka töötajate, tööks vajalike rakenduste ning serverite hulk, muutub aina olulisemaks asutuste infosüsteemis muutustele reageerimisvõimekus, mis peab olema kiirem ja efektiivsem. Seetõttu, et haldamist vajavad infosüsteemid lähevad aina keerulisemaks ja arendatavate teenuste olulisus ja kasutajate hulk on suurem kui kunagi varem, on ootused infosüsteemi kvaliteedile ka kõrgemad. Sellises keskkonnas on selge, et mida rohkem on asutuse süsteemide eest vastutavatel töötajatel informatsiooni ja mida täpsem see on, seda lihtsam on infosüsteemides toiumuvatele muutustele reageerida.

Vanad tööriistad ei rahulda enam tänapäevaseid vajadusi ja aeg on käes leida nende asemele uuemad. Käesolev töö uurib voogtelemeetriat, mis on arendatud just eelpool mainitud keskkonnas parema seire loomise eesmärgiga ja toob näite selle rakendamisest TEHIK-u põhjal.

Antud töö eesmärgiks on uurida voogtelemeetria praktilist kasutatavust võrguseadmete seire teostamiseks, mis oleks kiirem, täpsem ja lihtsam kui seiresüsteem, mida kasutatakse TEHIK-us täna.

1 TEHIK seiresüsteemi nõudmised ja ootused

Tänastes IT-süsteemides on tavaline, et käivitatakse väga lühikeseks ajaks suure koormusega rakendusi, mis kasutavad arvestatavat võrgu ressurssi, aga töötavad kõigest mõnikümmend sekundit. Klassikalise võrguseire praktika järgi küsitakse võrguseadmetelt nende töö parameetreid iga mõneminutilise intervalli tagant, mis võib autori kogemuste kohaselt olla ühe- kuni kümneminutilise vahega, mis tähendab, et eelpool nimetatud rakenduste töö mõju avastamine on selliselt väga ebatõenäoline.

Lahenduseks on tänaseks välja töötatud voogtelemeetria, mis erinevalt klassikalisest lähenemisest, kus mingi intervalli tagant seadmest päritakse selle staatust, saab nüüd võrguseade ise oma staatuse andmeid pideva andmevoona seiresüsteemi saata.

Tulenevalt käsitletava asutuse infrastruktuuri nõuetest, seirevajadustest ja tänastest kitsaskohtadest, on koostatud lühike nimekiri ootuseid ja nõudmisi voogtelemeetria-põhise seirekeskkonna loomisele (Tabel 1).

Tabel 1. Voogtelemeetria-põhise seirekeskkonna nõuded.

Nõue	**Kirjeldus**
1.	Andmed, mida monitooringu tarkvaradesse saadetakse, on ajakohased ja täpsete ajatemplitega
2.	Lahendus suudab tuvastada ja reageerida olukordadele, kus andmete saatmine seire tarkvarasse on peatunud
3.	Valitud seireprotokoll peab andma väljundi, mille kaudu saab jälgida seadme staatust ja ressursi kasutust
4.	Üle võrgu liikuvad andmed peavad olema krüpteeritud
5.	Lahenduse loomiseks peab eksisteerima tarkvara, mis ei nõua lisa arendustööd, on vabavaraline ja tasuta
6.	Ülesande lahedus peab olema laiendatav ja seadmete tootja suhtes võimalikult neutraalne, et väikese vaevaga oleks võimalik lisada monitooringusse teiste tootjate seadmeid
7.	Töö lahendus peab võimaldama voogtelemeetria andmete import liidestust järgmistesse monitooringu tarkvaradesse: Observium, Zabbix ja Prometheus
8.	Infrastruktuuri nõuete järgselt peavad kõik rakendused töötama operatsioonisüsteemil Linux

2 Seiresüsteemi objektide kirjanduse ülevaade

Seiresüsteemid on asutuse infosüsteemide infrastruktuuri toimimise hindamise alus, selle tulemuseks on sisend erinevatesse tööprotsessidesse alates infrastruktuuri planeerimisest kuni teenuste kvaliteedi hindamiseni. Seire andmeid kasutavad oma igapäevases töös asutuse süsteemiadministraatorid, projektijuhid, rakenduste administraatorid, võrguadministraatorid, arendajad ja arhitektid. Igaüks neist soovib monitooringust saada vajalikke indikaatoreid oma tööülesannete täitmiseks ja probleemidele reageerimiseks.

Seire sisendiks saab olla igasugune mõõdetav indikaator, näiteks serveri temperatuur, võrgus olevate seadmete kogus, sisse logitud kasutajad või rakendusse tehtud päringute arv.

Hea seiresüsteem suudab ennast automaatselt seadistada vastavalt infrastruktuuris toimunud muudatustele, anda graafilise ülevaate süsteemi seisukorrast ning tänu trendidel põhinevatele alarmsüsteemidele teavitada võimalikest riketest enne, kui need juhtuvad.

Järgnevates peatükkides uuritakse seiresüsteemide rakendusi, protokolle ja nende tausta. Esimesena käsitletakse selles töös üht vanimat, kuid siiamaani populaarset seireprotokolli, mille puudused ja piirangud on viinud voogtelemeetria arenduseni.

2.1 Lihtne võrguhalduse protokoll ehk SNMP

Ajal, mil internet oli uus tehnoloogia ja võrguseadmete hulk kasvutrendis, tekkis vajadus neid seadmeid kaugelt hallata ja nende tööd jälgida. Sellel eesmärgil loodi esmalt *Simple Gateway Monitoring Protocol* (edaspidi SGMP), kuid see oli piiratud ainult ruuterite seire ja halduse teostamiseks ning ei katnud kõiki vajadusi, mida võrguhaldurid ootasid [1].

SGMP järglaseks sai lihtne võrguhalduse protokoll (*Simple Network Management Protocol*, edaspidi SNMP), mille arendus algatati 1988. aastal Interneti arhitektuuri nõukogu (*Internet Activities Board*, edaspidi IAB) poolt. Eesmärgiks oli luua lihtne, vähest ressurssi nõudev ja kasutajasõbralik võrguseadmete halduse ja seire protokoll [2]. Tänaseks arendab seda protokolli *The Internet Engineering Task Force* (edaspidi IETF) ning see põhineb kolmel dokumendil: RFC 1155 [3], RFC 1212 [4] ja RFC 1157 [5].

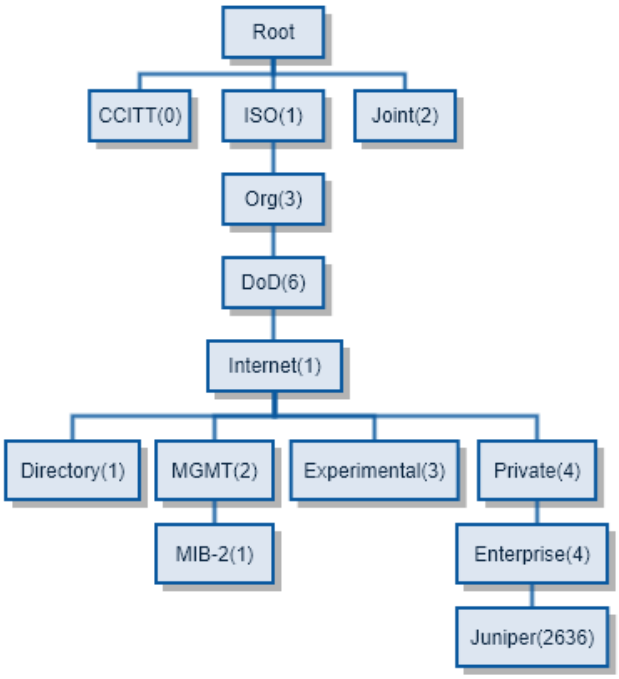
2.1.1 SNMP struktuur

SNMP on üles ehitatud kui klient-server teenus, kus server ehk *agent* on paigaldatud seadmesse ja vastab kliendi ehk *manager*-i päringutele, mis on üldiselt spetsiifiline halduse ja seire server *Network Management Station* (edaspidi NMS) [6].

SNMP koosneb kolmest komponendist:

1. Haldusprotokoll (*Management Protocol*), mille aluseks on RFC 1157. Selles dokumendis kirjeldatakse, kuidas SNMP tehniliselt töötab ning milline on selle protokolli arhitektuur [6].
2. Haldusinformatsiooni baas (*Management Information Base* edaspidi MIB). SNMP objektide andmebaas, mis põhineb RFC 1158 dokumentatsioonil [6].
3. Haldusinformatsiooni struktuur (*The Structure of Management Information* edaspidi SMI), millest on täpsemalt kirjeldatud RFC 1155 dokumendis. Sisuliselt defineerib see dokument, kuidas SNMP organiseerib oma objektide standardit, ja ei ole otseselt tehniline vaid pigem organisatoorne dokument [6].

Komponendid on omavahel seotud selliselt, kus MIB on seadme seadistuse objektide andmebaas ja iga objekt selles baasis omab unikaalset identifikaatorit, millel omakorda on kindel hierarhia (*Object Identifier Hierarchy*, edaspidi OID). OID-l on puukujuline struktuur (Joonis 1) ja see on ASN.1 kodeeringus. Selle ülesehitus on kirjeldatud SMI dokumentatsioonis. OID-d on kõik unikaalsed ja nende määramise eest vastutab iga seadme tootja oma MIB andmebaasides ise [6].



Joonis 1. OID Struktuur [6], [7].

Võrguseire jaoks tähtsad OID-d algavad 1.3.6.1. identifikaatoriga. Võtame näiteks pikema OID, mis juhatab selles töös olulise tootja MIB identifikaatorini, ja seletab lahti praktilise näitega, kuidas OID-d on struktureeritud. Näite OID on 1.3.6.1.4.1.2636, mis kuulub Juniperi MIB-i.

- (1).3.6.1.4.1.2636 – esimene identifikaator kuulub Rahvusvahelisele Standardimisorganisatsioonile (*International Organization for Standardization* ehk ISO).
- 1.(3).6.1.4.1.2636 – teine number grupeerib organisatsioonide identifikaatoreid.
- 1.3.(6).1.4.1.2636 – kolmas number ütleb, et OID-d haldab selle näite järgi Ameerika Ühendriikide Kaitseministeerium kuid olukord on pisut keerulisem, kuna järgmise identifikaatori kohaselt liigub vastutus selle alt ära ja edaspidi haldab antud alamkategooriat Interneti Kogukond (*Internet Community*) [3].
- 1.3.6.(1).4.1.2636 – neljas number ütlebki, et see alamkategooria kuulub Interneti Kogukonnale.
- 1.3.6.1.(4).1.2636 – selles kategoorias on tegelikult mitu alamharu, nendeks on *Directory*, *MGMT*, *Experimental* ja *Private*, viimase all hallatakse tootjate spetsiifilisi MIB andmebaase ja oluline on märkida, et MGMT haru alla kuulub MIB-2 andmebaas, mis on standardne MIB ja peab olema rakendatud kõigis võrguseadmetes [6].

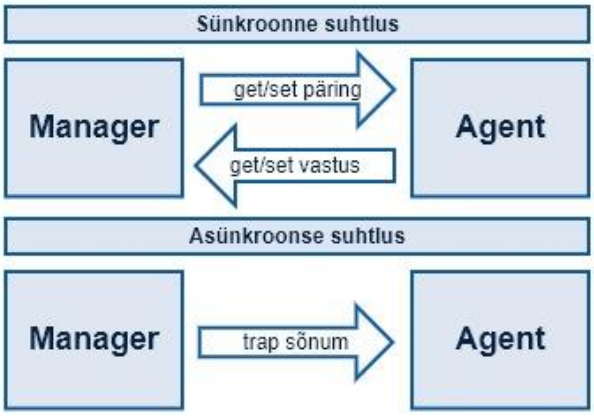
- 1.3.6.1.4.(1).2636 – see haru võimaldab muu hulgas eraettevõtetel registreerida oma MIB andmebaase. Viimane number on 2636 ja kuulub Juniper-le.

2.1.2 SNMP kommunikatsioon

Antud protokoll võimaldab kasutada nelja operatsiooni, milleks on *get*, *get-next*, *set* ja *trap*. *Get* operatsioon tagastab kliendile küsitud parameetri väärtuse, näiteks võrguseadme temperatuuri. *Get-next* tagastab järgmise väärtuse seadme MIB andmebaasist alustades väärtusest, mis on varasemalt teada. Näiteks, kui eelnevalt on küsitud võrguseadme esimese liidese staatust, siis *get-next* annab järgmise liidese staatuse väärtuse. Seda operatsiooni kasutades on võimalik agendi käest saada kogu seadme staatuse informatsiooni läbides ükshaaval kogu andmebaasi, kuid selline tegevus on üsna ressursi nõudlik ja on üheks miinuseks SNMP puhul [8].

Antud protokolli üks peamisi seire teostamisviise on nn *trap* sõnumid, mis on oma loomult asünkroonsed ja saadetakse enamasti kas mingi intervalli tagant teatud parameetri raporteerimiseks, mingi kriitilise sündmuse järel või mõne muu muutuse tagajärjel [1]**.**

Agendi ja *Manageri* suhtlust saab pidada kahel viisil: sünkroonselt või asünkroonselt (**Error! Reference source not found.**). Mõlema suhtluse puhul kasutatakse võrgu transpordi protokollina *User Datagram Protocol-i* (edaspidi UDP), mis tähendab, et SNMP suhtluse käigus tulenevalt UDP omadustest võivad osa sõnumeid kaduma minna [8].



Joonis 2. SNMP sünkroonne ja asünkroonne suhtlus

Sünkroonne suhtlus käib *polling* mehhanismi abil ja seda kasutatakse kõigi operatsioonide puhul peale *trap* operatsiooni, kus *manager* küsib agendi käest OID muutuja väärtust ja *agent* vastab muutuja väärtusega. *Trap* operatsiooni puhul saadab *agent managerile* teateid kas perioodiliselt või muutuste tekkimisel [1].

SNMP on üle kolmekümne aasta vana protokoll ja endiselt laialdaselt kasutusel. Selle aja jooksul on seda protokolli arendatud ja täiendatud uute versioonidega, kuid oma arhitektuurilt on ta samasugune nagu siis, kui see välja tuli. Ilmselt ei kao see protokoll IT maastikult ja administraatorite tööriistakastist veel palju aastaid, kuid suuremaid muutusi ei maksa selles ka oodata, kuna aktiivsemas arenduses on uuemad tehnoloogiad ja protokollid, millest tuleb juttu järgmistes peatükkides.

2.2 Voogtelemeetria

Voogtelemeetria tähendab pidevat andmete saatmist kaugelt asuvast seadmest neid koguvasse teenusesse seire eesmärgil [8]. See on sisuline erinevus SNMP päringute põhisest seirest, kus andmeid jälgiv süsteem peab tegema, kindla intervalli tagant, pidevaid päringuid seires olevale seadmele. Need päringud võivad olla väga suured ja raskelt töödeldatavad. Lisaks sellele võib päringuid tegevaid kliente olla rohkem kui üks, mis tähendab SNMP kontekstis, et iga päringu vastuseks peab seade tegema tööd ja kui tegemist on suuremahulise vastuse töötlemisega, võib see praktikas tähendada, et seire tegevus kasutab ära suure hulga seadme ressursist. Sellist probleemi üldiselt voogtelemeetriat pakkuvate protokollidega ei ole, kuna erinevad kliendid saavad sama eeltöödeldud vastuse [9].

Voogtelemeetria kogumist võib võrrelda tuttavate *Trap* sõnumitega, kuid erinevalt nendest ei ole seda toetavad protokollid realiseeritud kasutades UDP transpordi protokolli ja ei ole häiritud sellest tulenevatest ebakindlustest andmete saatmisega [8].

Voogtelemeetria koosneb üldises mõttes kahest komponendist, milleks on transpordi protokoll ja andmemudel. Autor uurib oma töös kahte voogtelemeetriat toetavat protokolli, milleks on *Network Configuration Protocol* (edaspidi NETCONF) ja *gRPC Network Management Interface* (edaspidi gNMI). Andmete modelleerimise keelena kasutavad üldiselt antud protokollid *Yet Another Next Generation* (edaspidi YANG) mudeleid, millest räägib järgmine peatükk.

2.3 YANG mudelid

YANG on tüübitud andmete modelleerimiskeel, mille eesmärgiks on kirjeldada, kuidas andmed peaksid välja nägema.

See keel sai alguse RFC 6020 dokumentatsioonist ja loodi IETF töörühma poolt aastal 2010 NETCONF protokolli andmete modelleerimiseks [10]. Oluline on märkida, et YANG kasutab oma andmete kodeeringuks laiendatavat märgistuskeelt (*Extensible Markup Language* edaspidi XML) alates versioonist 1.0, kuid hilisema uuendusena lisandus ka *JavaScript Object Notation* (edaspidi JSON), mis on mõnevõrra kompaktsem kui XML [11].

Kuigi see keel sai alguse vajadusest modelleerida võrguseadmete andmeid, ei ole see piiratud ainult antud andmete kasutuse valdkonnas, vaid sobib igasuguste andmemudelite koostamiseks [8].

YANG-il on hästi loetav ja lihtne puu-sarnase ülesehitusega struktuur ja tänu oma modulaarsele disainile on see laiendatav ja võimaldab kasutada olemasolevate mudelite mooduleid ja tüüpe uute mudelite loomiseks [10].

Kuna YANG-i eesmärk on pakkuda sarnaselt NETCONF-ile lihtsat ülemineku võimalust SNMP-lt uuematele protokollidele, on sellel tugi tõlkida SNMP SNMIv2 põhiseid MIB baase otse YANG mudeliteks, kuid mitte teistpidi.

2.3.1 YANG mudelite struktuur

YANG mudelite väikseimaks objektiks on *leaf* mis hoiab ainult ühte tüübitud andme väärtust, nagu *string*, *integer*, *boolean* vms; mudelis on ka leaf-i kirjeldus ehk vabas vormis tekst, mille eesmärgiks on anda mudeli kasutajale infot selle väärtuse sisu kohta [10].

Leaf-e saab esitada kas ühe väärtusena või jadana, kus jada koosneb üksikutest leaf -idest (Joonis 3 ja Joonis 4).

```
leaf auto {
    type string
    description „Auto mudel“
}
```

Joonis 3. Leaf näide

```
Leaf-list auto-margid {
    type string
    description „Automarkide loend“
}
```

Joonis 4. Leaf-list näide

YANG rakendab kontrolli tüübi definitsiooni kohta. Mis tähendab, et kui mudelis on määratud, et väärtus on string, siis selle *leaf*-i väärtuseks ei saa määrata midagi muud [10].

YANG võimaldab luua ka kasutajate spetsifitseeritud andmetüüpe ja nendele tüüpidele luua kontrollmehhanisme, mis tähendab, et YANG-i kasutaja saab näiteks teha tüübi IP (*Internet Protocol*) ja määrata viisi, kuidas seda tüüpi kontrollitakse, et mudeli kasutaja ei saaks antud väärtuseks midagi muud kasutada kui IP (*Internet Protocol*) tüüpi. Seda on võimalik antud keeles rakendada regulaaravaldistega [12].

YANG võimaldab grupeerida *leaf*-e konteineritesse, kusjuures konteinerid võivad sisaldada nii *leaf*-e kui ka teisi konteinereid (Joonis 5).

```
container sõiduvahendid {
    leaf kirjeldus {
        type string;
        description „Erinevad sõiduvahendid“
    }
}
```

Joonis 5. Näide: YANG konteinerid

Lisaks ülalpool toodud näidetele andmemudeli struktureerimisvõimalustest, on YANG-il palju teisi elemente, mis muudavad selle keele väga paindlikuks. Nendeks on näiteks valikute kirjeldused, staatuse kirjeldused, nimekirjad jt. mida pole kõiki võimalik autoril oma töös kirjeldada. Kuid Need on lihtsasti leitavad selle modelleerimiskeele RFC dokumendist [10].

2.4 NETCONF

2002. aastal pidas IAB võrguhalduse teemadel tööseminari, mille eesmärgiks oli jätkata olulist dialoogi võrguoperaatorite ja protokolliarendajate vahel ning juhendada IETF-i võrguhaldusega seotud tuleviku tööde osas. Selle käigus leiti hulk probleeme, millega võrguoperaatorid igapäevaselt pidid tegelema ning mille jaoks oli vaja leida paremad tööriistad ja lahendused. Enamik probleeme tulenes standardite puudumisest ja võrguseadmete vahelistest erinevustest [13].

IETF alustas NETCONF-i loomisega 2006. aastal vastusena IAB tööseminaril ülestõstatatud probleemidele [13]. Selle peamiseks ülesandeks oli standardiseerida võrguseadmete haldus ja seiresüsteem, mis annaks parema ülevaate seadme tööst kui seda suutis teha SNMP.

Tulemuseks loodi protokoll, millega on võimalik seadmeid hallata ja teostada seiret ning tänu paljudele vabavaralistele moodulitele eri keeltes on selle kasutamine arendajatele tehtud võimalikult lihtsaks. Enamik uusi seadmeid on tänaseks NETCONF-i juba kasutusele võtnud [8].

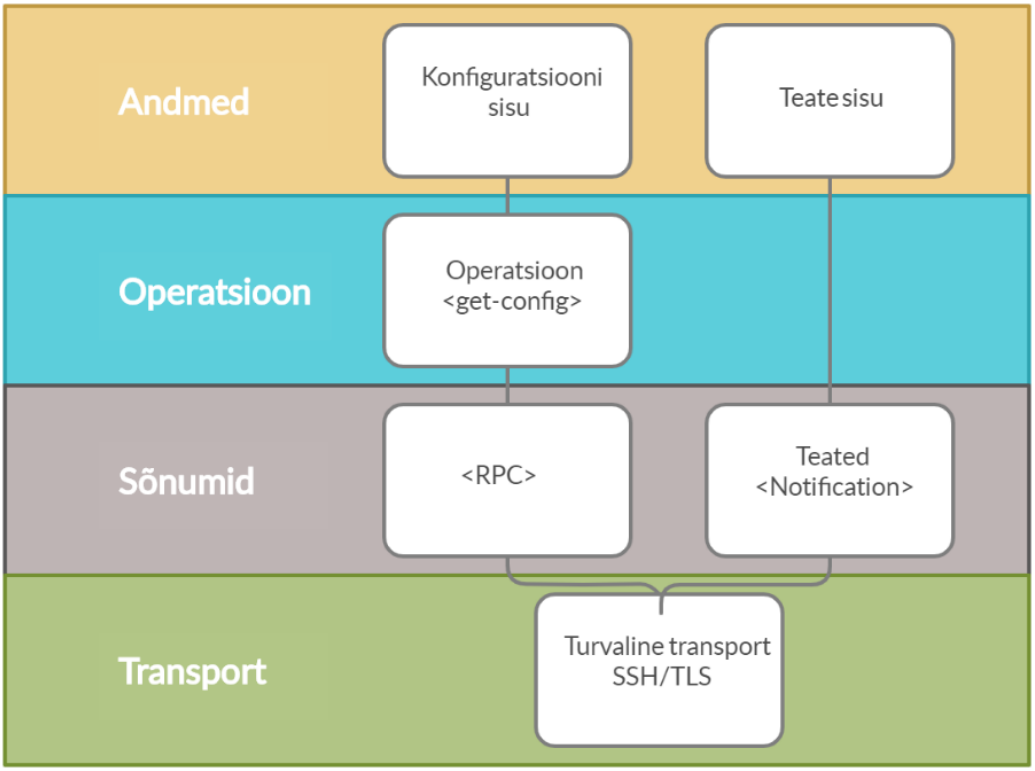
Siiski kõik probleemid kohe lahendust ei leidnud. Kuna andmete modelleerimiseks selleks ajaks veel ühtset lahendust ei olnud, tekkis olukord, kus seadme tootjad hakkasid arendama oma modelleerimiskeeli, nagu näiteks Juniperi *Data Definition Language* [14], selle probleemi lahenduseks loodi YANG.

2.4.1 NETCONF protokolli ülesehitusest

NETCONF on klient-server topoloogiaga kaugprotseduurikõne (*Remote Procedure Call*, edaspidi RPC) protokoll, mis tähendab, et sellega käivitatakse teises seadmes, olgu see siis mõni server või antud olukorras võrgu seade, mõni programm või programmi meetod. Kliendi ja serveri vahel saab eksisteerida ka ülekande teenust pakkuv klient-server tüüpi süsteem, mille otstarve on vahendada suhtlust nende kahe vahel [8].

NETCONF-i kasutamine algab sessiooni loomisega, mille käigus saadetakse tervitussõnum mõlemale osapoolele, mille sisu deklareerib mõlema osapoole protokolli versiooni ja osapoolte oskused; server annab kliendile teada, milliseid lisafunktsioone ta kliendile pakub. Suhtluse alustab alati klient [8], [15].

Kliendi ja serveri vaheline suhtlus töötab üle erinevate turvaliste kommunikatsiooni protokollide ja kasutab andmete kodeeringuks XML kirjelduskeelt. Et paremini aru saada NETCONF-i tööpõhimõttest, võib seda eristada neljaks tasemeks, nagu kirjeldatud RFC dokumendis (Joonis 6) [16].



Joonis 6. NETCONF-i neli taset. Loodud RFC näite põhjal [16]

- Esimene tase kirjeldab protokolli sõnumite edastuseks kasutatavat transporti. Nendeks kanaliteks võib olla SSH, TLS või HTTP.
- Teine tase kirjeldab sõnumi tüüpi (nt RPC, RPC Reply või Notification) Kõik sõnumid on kodeeritud XML formaadis, mis võimaldab keerukat hierarhilist andmete kirjeldust.
- Kolmas tase kirjeldab käivitatud operatsiooni (nt get-config või edit-config).
- Neljas tase sellel protokollil on sisu, näiteks konfiguratsioon, mida muudetake või teated, mida klient saadab serverile.

Iga RPC käsu sõnumile pannakse kaasa kohustuslik sõnumi id, mille abil sõnumid ja vastused omavahel kokku viiakse ning XML kujul edastatud meetodi nimi koos selle parameetritega.

NETCONF-i üks kasulikemaid funktsionaalsusi võrguseire vaates on selle võime saata voogtelemeetriat kasutades tellimus süsteemi (S*ubscription*). Lihtsustatult võib vaadelda seda kui RPC päringut, mille vastus kunagi ei lõppe [8]. Päring on tellimus ja vastus päringule on pidev andmevoog.

2.5 OpenConfig

OpenConfig on töörühm, kes ühendab mitteformaalses koostöös maailma suurimad võrguoperaatorid, nende hulgas on sellised ettevõtted nagu Microsoft, Google, Netflix, LinkedIn, Apple, Baidu jpt. Selle töörühma eesmärk ja ajend on asendada SNMP uuema tehnoloogiaga, mis võimaldab kasutada ühtset andmemudelit üle kõigi tootjate seadmete, muutes võrguhalduse lihtsamaks ja arusaadavamaks [17], [18].

See töörühm on loonud võrdlemisi lühikese aja jooksul kaks kasulikku tehnoloogiat, milleks on gNMI protokoll ja OpenConfig YANG mudelid [19].

OpenConfig eesmärk on muuta IP võrkude infrastruktuur dünaamiliseks ja programmaatiliseks, kasutades *Software Defined Networking* lähenemist ja mille keskseks fookuseks on välja töötada tootja-neutraalne ja mudeli-põhine võrguseadmete haldus [8], [18].

Tänaseks on kaks suuremat gruppi, kes arendavad võrguhalduse eesmärgil YANG mudeleid. Need on OpenConfig, mille loomisega alustati 2014. aastal ning eelpoolmainitud IETF, kes alustas oma mudelite loomist ja standardit 2010 RFC 6020 dokumendiga [10], [18].

Selles töös ei võrrelda IETF ja OpenConfig mudelite sarnasusi ja erinevusi, kuna see läheb autori skoobist välja ja nõuaks pikemat analüüsi. Lühidalt võib aga välja tuua selle, et mõlemad kasutavad YANG-i, töötavad samadel protokollidel ja erinevused on pigem ideoloogilised, lähemalt saab lugeda nende erinevusest antud raamatust [8].

OpenConfig grupi üks olulisemaid saavutusi ja eesmärke SNMP asendamiseks on voogtelemeetria arendamine, mis töötab üle gNMI liidese. Antud liides ise on loodud kasutades Google-i arendatud *gRPC Remote Procedure Call* (edaspidi gRPC) protokolli [8], gRPC nimi ise on rekursiivne ja sisaldab viidet oma akronüümile, mistõttu polegi kindel mida see tegelikult tähendab [20].

Voogtelemeetria on oluline uuendus võrguseadmete monitooringus, kuna võimaldab väga tihedat andmevoogu, mis annab oluliselt parema ülevaate võrgu liiklusest ja seadmete töös. Tänu gRPC tehnoloogias kasutatavale Google-i arendatud protokolli puhverandme kodeeringule on voogtelemeetria üle gNMI vähest võrgu ressurssi ja arvutusvõimsust nõudev. See on oluline parendus võrreldes NETCONF-i XML kodeeringuga [21], [22].

2.6 gNMI

gNMI võimaldab automaatset võrguseadmete seadistust ja teeb oluliselt lihtsamaks võrguadministraatorite töö. Nagu eelpool mainitud, on üks selle suurimaid eeliseid selle võime edastada ressurssi kokkuhoidvat voogtelemeetriat, kasutades andmete modelleerimiseks YANG mudeleid ja on uuem alternatiiv NETCONF-ile, jagades samu eesmärke, kuid kasutades selleks erinevaid tehnoloogiaid [23]. See on võrguseadmete seire- ja seadistusliides, mis on arendatud gRPC protokolli raamistikule. Antud tarkvara raamistik võimaldab arendajatel kirjutada standardse aplikatsiooni liidesega RPC programme. Seda tehnoloogiat kasutatakse tänapäeval väga palju mikroteenuste loomisel [20]. gRPC võimaldab kasutajal kodeerida andmevoogu, kasutades selleks erinevaid võimalusi nagu JSON või kõige olulisemana protokolli puhvrit (*protocol buffer*) kodeeringut [24].

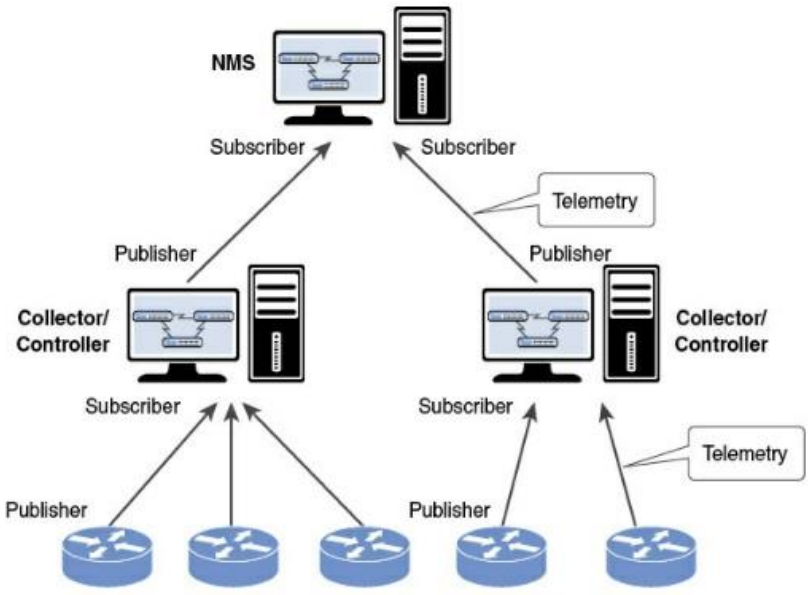
Google-i protokolli puhver võimaldab serialiseerida suurt hulka struktureeritud andmeid. See on sarnane XMLi või JSON-iga, kuid kasutab selleks vähem ressursse, kuna vajab andmete edastamiseks väiksemat andmehulka [22].

gRPC raamistik toetab erinevaid programmeerimiskeeli nagu C++, Java, Python ja isegi PHP-d ja Ruby-t [25]. Tänu sellele on gNMI-i kasutamine võimalik samuti kõigis gRPC-d toetavates keeltes.

Antud RPC protokoll võimaldab kasutada nelja operatsiooni, milleks on:

- *CapabilityRequest* – Sarnaselt NETCONF *hello* sõnumile vastab gNMI kliendile protokolli ja mudelite versiooni ja seadmes toetatud kodeeringu, nagu näiteks JSON või protokolli puhver, infoga [8].
- *GetRequest* – On mõeldud serverilt väikeste andmehulkade, näiteks YANG leaf-i väärtuse või mingi mudeli osa staatuse küsimiseks. See ei sobi suurte andmehulkade küsimiseks [8].
- *SetRequest* – See operatsioon võimaldab seadistada väärtusi võrguseadmes ja seda koheldakse kui transaktsioonilist operatsiooni, mis tähendab, et kui antud käsk mingil põhjusel ebaõnnestub, siis automaatselt laetakse seadmes eelnev seis [8].
- *SubscribeRequest* – Autori töö jaoks kõige olulisem osa gNMI protokollist. OpenConfig grupp investeeris palju aega ja pani rõhku selle osa arendamisele, kuna see on mõeldud voogtelemeetria või suurte andmepäringute edastamiseks [8].

gNMI *SubscribeRequest* on mõeldud voogtelemeetria edastamiseks. Klient saab küsida korraga mitme YANG mudeli puu väärtusi; vastuseks on pidev andmevoog. Oluline on, et selle andmevooga saavad liituda ka teised kliendid, kuid erinevalt SNMP-st ei ole vaja selle tegevuse käigus seadmel andmeid iga kliendi jaoks lugemiseks ette valmistada [8].



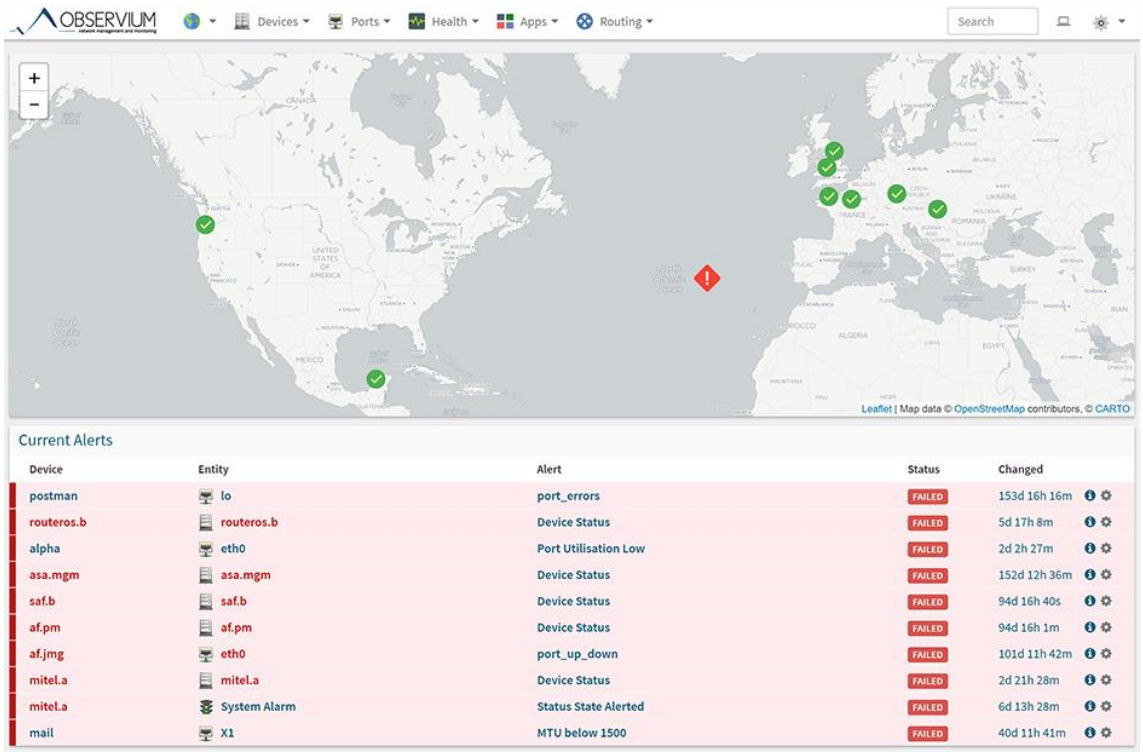
Joonis 7. Voogtelemeetria arhitektuuri näide, pilt võetud raamatust [8]

2.7 Observium

Observium on võrguseire spetsiifiline tarkvara, mis selle kodulehel välja toodult, on arendatud võrguinseneride ja süsteemiadministraatorite poolt, ja disainitud just oma kasutajate ootusi arvestades. See on tasuta kättesaadav vabavaraline tarkvara, millele pakutakse lisaks kahe-tasemelist tasulist versiooni. Tasulised versioonid lisavad mitmeid funktsionaalsusi ja kasutajatuge vastavalt tasemele [26].

Observium on loodud võrgu monitooringu eesmärgil, kuid see pole antud tarkvara ainuke kasutus, kuna võimaldab ka võrguseadmete konfiguratsiooni haldust tänu integratsioonile RANCID haldustarkvaraga.

Observiumi üheks eeliseks on selle kasutuslihtsus ja intuitiivne veebi-põhine kasutaja keskkond, mis pakub võrguseire graafikuid, kasutades selleks Round Robin Database andmebaase ja interaktiivsete kaartide koostamist (Joonis 8) [26].



Joonis 8. Observiumi interaktiivse kaardi näide [26]

Antud rakendusel on mitmeid funktsionaalsusi, mis on ühised teiste samalaadsete tarkvaradega, milleks on näiteks teavituste saatmine ja operatsioonisüsteemide monitooring, kasutades selleks agent-rakendust, mis koosneb erinevatest väikestest skriptidest ja tuleb paigaldada seires olevasse serverisse [26].

Observiumi suurim eelis, aga samas ka suurim miinus on see, et antud seire tarkvara toetab agent-skriptide kõrval ainult SNMP protokolli, kuid ta teeb seda väga hästi. Võrguseadmete seiresse panemine on lihtne ja tänu selle laialdasele MIB andmebaaside kogule ei ole vaja kasutajal ise MIB andmebaaside haldamisega tegeleda [26].

Suurimaks eeliseks loeb Observium ise oma automatiseeritud seadmete seiresüsteemi, kus tarkvara suudab otsida võrgust seadmed üles ja need monitooringusse panna, tuvastades nende seadmete tootja ja mudel, kasutades SNMP MIB-de põhist identifitseerimist [26].

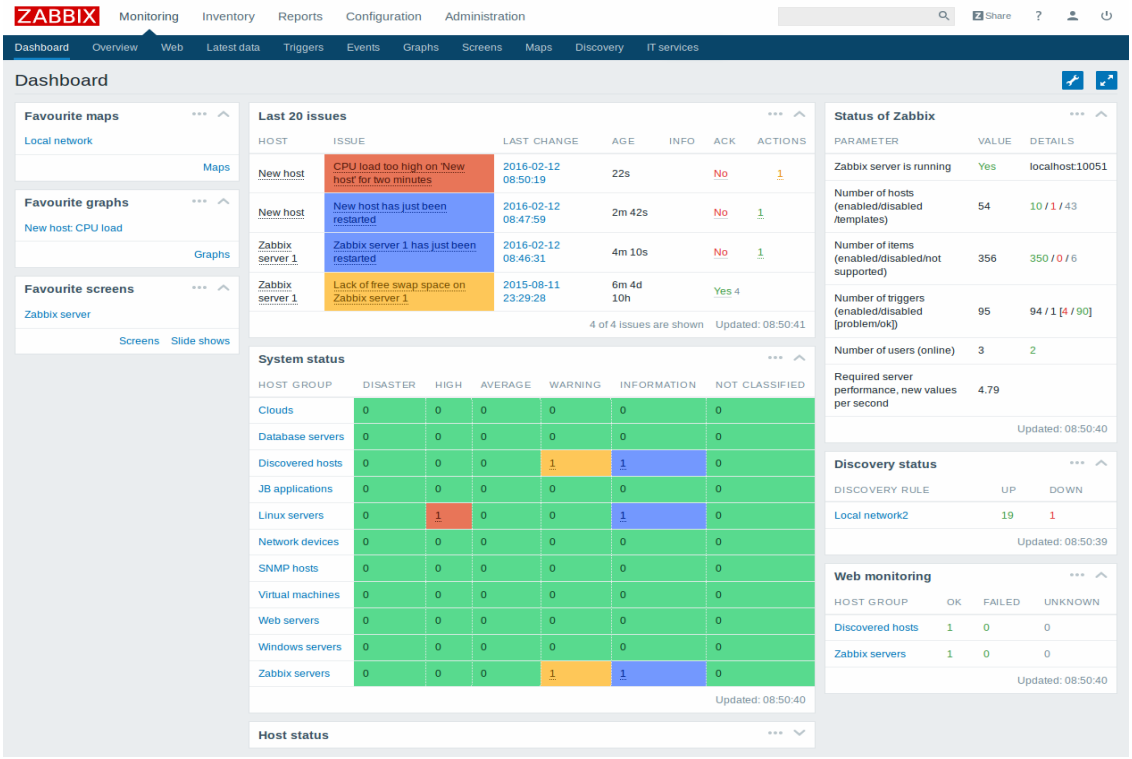
2.8 Zabbix

Zabbix on üks populaarsemaid vabavaralisi monitooringu tarkvarasid, mis tänapäeval on üle maailma kasutusel. Kuigi see on vabavaraline tarkvara, ei peida selle arendajad ühtegi funktsionaalsust tasulise versiooni taha ega paku tasulist versiooni, kuid kliendi vajaduse järgi on võimalik kasutajatoe leping sõlmida ja arendust tellida [27].

Oma kahekümne aastase arenduse jooksul on Zabbix olnud innovatiivne ja uusi versioone ilmub iga aasta. Selle arenduseks on kasutatud erinevaid keeli. Iga keel on valitud selle parima sobivuse järgi mingi ülesande täitmiseks, näiteks C-s on kirjutatud selle tuumikserver, Go on kasutusel agentide jaoks ja PHP-s on kirjutatud Zabbix-i veebiliides. Andmebaasina on võimalik kasutada nii MySQL-i, PostgreSQL-i ja Oracle andmebaase [28].

Zabbix on väga mitmekülgne seiresüsteem, see võimaldab jälgida nii andmebaaside, operatsioonisüsteemide, veebiteenuste kui ka võrguseadmete tööd kasutades SNMP-d kui ka virtualiseerimiskeskkondade staatust.

Lisaks kõigele sellele võimaldab antud tarkvara lihtsa vaevaga laiendada oma agent-programme, mis tähendab, et isegi kui on tarvis seiresse lisada mõni teenus, mida tavaliselt Zabbix jälgida ei suuda, on see siiski lihtsa vaevaga võimalik [29].



Joonis 9. Zabbix seire vaade, näide võetud toote koduleheküljelt [30]

Võrreldes oma konkurentidega on selle tarkvara selge eelis tema pikaajaline ajalugu, laialdane valik funktsionaalsusi ja suur kasutajate baas, mistõttu on Zabbix-i kohta võimalik pikema otsimiseta leida mitmeid raamatuid, blogipostitusi ja abi foorumitest.

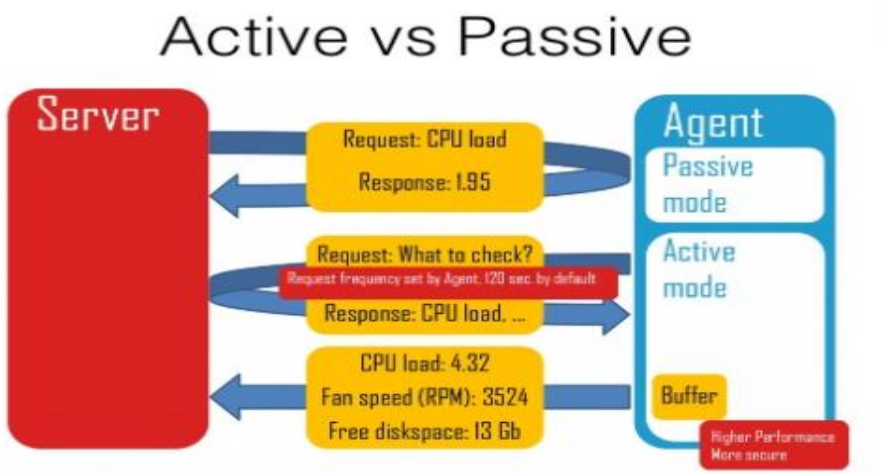
Sarnaselt Observiumiga pakub Zabbix võimalust kasutada SNMP protokolli võrguseadmete monitooringuks, kuid ei paku MIB andmebaaside kogu ja ei oma kõiki samu funktsionaalsusi, näiteks ei paku Zabbix võrguseadmete konfiguratsioonihaldust.

Sellegipoolest võimaldab Zabbix samuti koostada kaarte, graafikuid ja korrelatsioone, kuid erinevalt Observiumist, tuleb kasutajal need Zabbixis ise seadistada. Zabbixi eelis teiste, samalaadsete toodete ees, mida TEHIK kasutab, on automaatne serverite monitooringusse registreerimine, kasutades selle *agent*-rakendust [28].

Seiresse registreerimise järel suudab Zabbix hakata automaatselt koguma selle serveri jõudlusparameetreid nagu CPU kasutus, kõvaketta täituvus kõigil partitsioonidel, teenuste töö ja protsesside arv jpm. Lisaks nimetatule võimaldab Zabbix lihtsat viisi

laiendada agendi võimekust kasutaja poolt loodud skriptide abil, mis võimaldab kasutajal jälgida oma teenuseid, mis serveril on käivitatud.

Zabbixi agent ise võib töötada kahel viisil: passiivselt või aktiivselt (Joonis 11). Passiivne agent ootab serverilt päringuid ja saadab serverile seireandmeid ainult siis, kui neid küsitakse. Aktiivne agent saadab serverile ise seireandmeid talle ette antud intervalli tagant. Zabbixi agent võib töötada mõlemas seadistuses korraga ja suudab eristada milliseid andmeid ta peab saatma aktiivselt ise ja milliste andmete jaoks ta ootab serveri päringut [28].



Joonis 10. Zabbixi aktiivne ja passiivne agent. pilt võetud Zabbix-i koduleheküljelt [31]

2.9 Prometheus ja Grafana

Prometheus on vabavaraline seire- ja teavitusrakenduste tööriistade komplekt, arendatud SoundCloud ettevõttes alates 2012. aastast. Prometheus on muutunud üha populaarsemaks tööriistaks just pilverakenduste seire teostamiseks ja on integreeritud paljudes pilvesüsteemides [32].

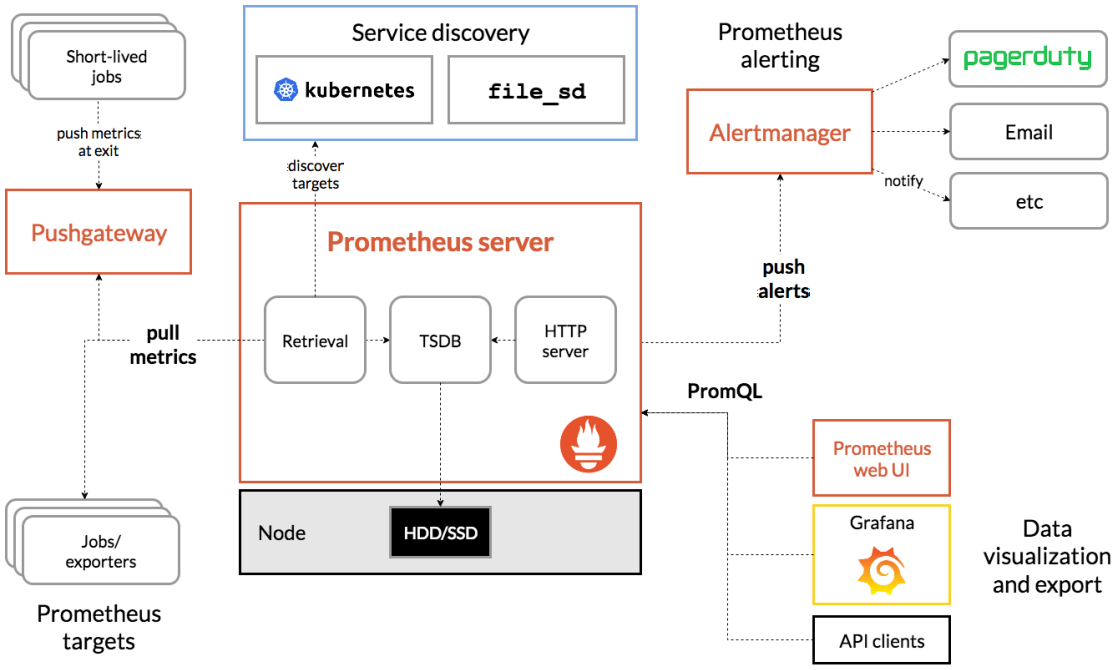
Prometheus-i teeb eriliseks selle andmemudel, aegridade andmebaas, visualiseerimise võimalused ja uuenduslik efektiivne andmesalvestus. Sellele tarkvarale on loodud palju integratsioone ja tarkvara teeke, mistõttu on antud tarkvara väga tihti esimene valik arendajatel oma rakenduste töö seire teostamiseks [32].

Prometheus-i olulisemad funktsionaalsused on [32]:

- Multidimensionaalne andmemudel, mis võimaldab salvestada aegseeria andmeid selle meetrika nime ning lisaks võtme- ja väärtuste-põhisena.
- Juurde on arendatud PromQL andmebaasi keel, mis võimaldab eelnimetatud andmemudelite põhjal salvestatud andmeid andmebaasist pärida väga sarnaselt SQL päringutele.
- Aegridade andmeid kogutakse keskselt kasutades selleks *pull*-mehhanismi, mis tähendab, et Prometheus server käib andmeid küsimas kas *Pushgateway* süsteemilt või *Explorer* kliendilt.
- Prometheus võimaldab asünkroonselt saata andmeid selle Pushgateway vaherakendusse, mis annab võimaluse arendajatel saata seireandmeid ka süsteemidest, kus muul juhul andmeid küsimas ei ole võimalik käia.
- Prometheus ei sõltu teistest salvestusmehhanismidest nagu kesksed andmebaasid. See on täiesti autonoomne, mis teeb paigalduse ja halduse lihtsamaks.
- Tänu teenuste avastusmehhanismile (*Service Discovery*) suudab Prometheus seiresse võtta suurel hulgal teenuseid ilma mingi seadistuse vajaduseta.

Prometheus koosneb paljudest komponentidest (Joonis 11), millest märkimisväärsed on järgmised [32]:

- Põhikomponent on Prometheus server, mis kogub andmed kokku ja salvestab need aegridade andmebaasi
- Klient-teegid**,** mille abil on võimalik oma rakendustesse Prometheus**e** seire lisada.
- Saatmis bastion (*Pushgateway*), kuhu rakendused saavad oma seire parameetrid saata. Selle kasutuseesmärk on koguda lühikese käivitusega serveri vabade teenuste meetrikaid, näiteks Kuberneteses käivitatavad lühikese eluga konteinerid [33] [34].
- *Exporter* klient rakendus, mis annab võimaluse serveris töötavate teenuste seire teostamiseks, sh serveri oma staatuse ja töö seire nagu CPU kasutus, või ketta täituvus.
- Häirete juhtsüsteem (*Alertmanager*), mis koordineerib teadete saatmist ja häirete tuvastamist.
- Palju integratsiooni- ja tugitööriistasid.



Joonis 11. Prometheuse arhitektuuri ülevaade, pilt võetud toote koduleheküljelt [32].

Prometheus enamasti paigaldatakse koos Grafana visualiseerimistarkvaraga, mis võimaldab kasutajatel lihtsa vaevaga luua graafilisi vaateid oma seire andmete kohta [35].

Antud tarkvara arendab Grafanalabs alates 2014. aastast ja see ettevõte ei ole otseselt seotud Prometheuse-ga, kuid sobivad omavahel niivõrd hästi kokku, et enamasti kui räägitakse Prometheusest siis Grafana olemasolu on eelduslik.

Grafana võimaldab visualiseerida aegridade andmeid erinevatest allikatest ja ei ole tihedalt seotud Prometheusega. Sellel tarkvaral on lai valik pistikprogramme andme allikatega ühendamiseks, nende hulgas ka liides eelpool kirjeldatud Zabbixiga ja sobib hästi keskseks visualiseerimise platvormiks [35].

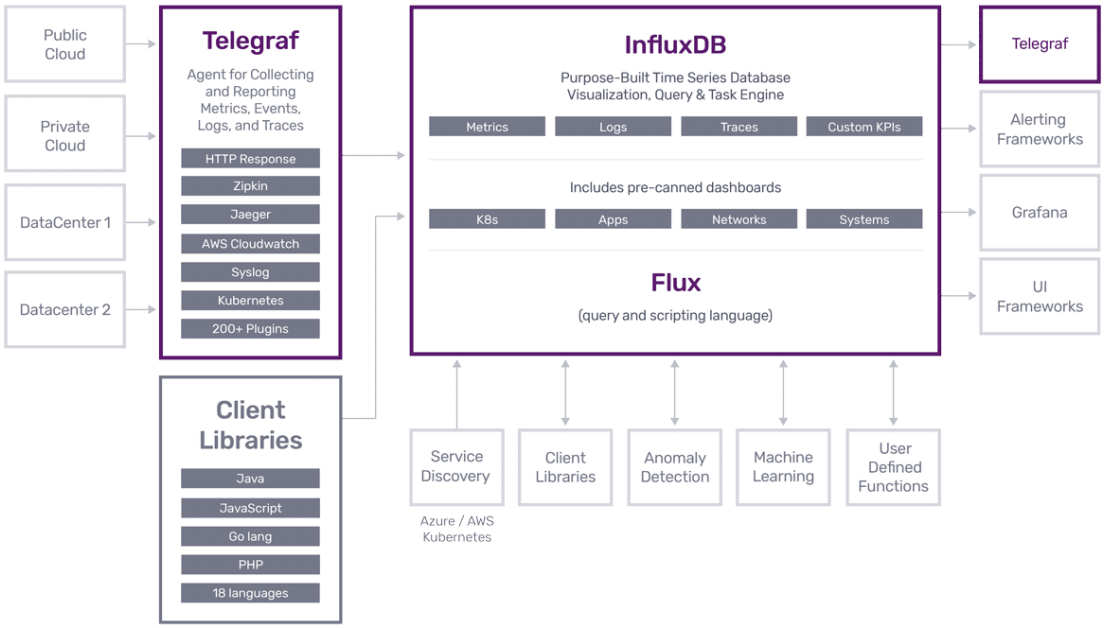
Joonis 12. Grafana seire vaade Prometheuse sisendi andmetega, näide võetud Grafana koduleheküljelt [36].

Grafana on tänaseks üks populaarsemaid visualiseerimise tarkvarasid ja kasutusel suurtes ettevõtetes nagu PayPal, Ebay ja Intel [35].

2.10 Telegraf

Telegraf on vabavaraline pistikprogrammidel põhinev serverteenus, mille eesmärk on koguda seire andmeid teenustest, andmebaasidest ja pilveteenustest. Oma loomult on ta üsna sarnane Prometheuse Exporterile, kuid erineb sellega, et on lihtsasti laiendatav pistikprogrammidega ja nendest on sellel projektil võrdlemisi suur valik [37].

See on üks InfluxData toodetest ja eelkõige loodud olema InfluxDB sisendiks, kuid töötab sama hästi Prometheuse sisendina, kuna oma pistikprogrammide valikus on tal ka Prometheuse Exporter. InfluxDB ise on alternatiiv Prometheusele, pakkudes väga palju sarnaseid tööriistu ja funktsionaalsust [32].



Joonis 13. Telegraf arhitektuur, pilt võetud toote koduleheküljelt [37]

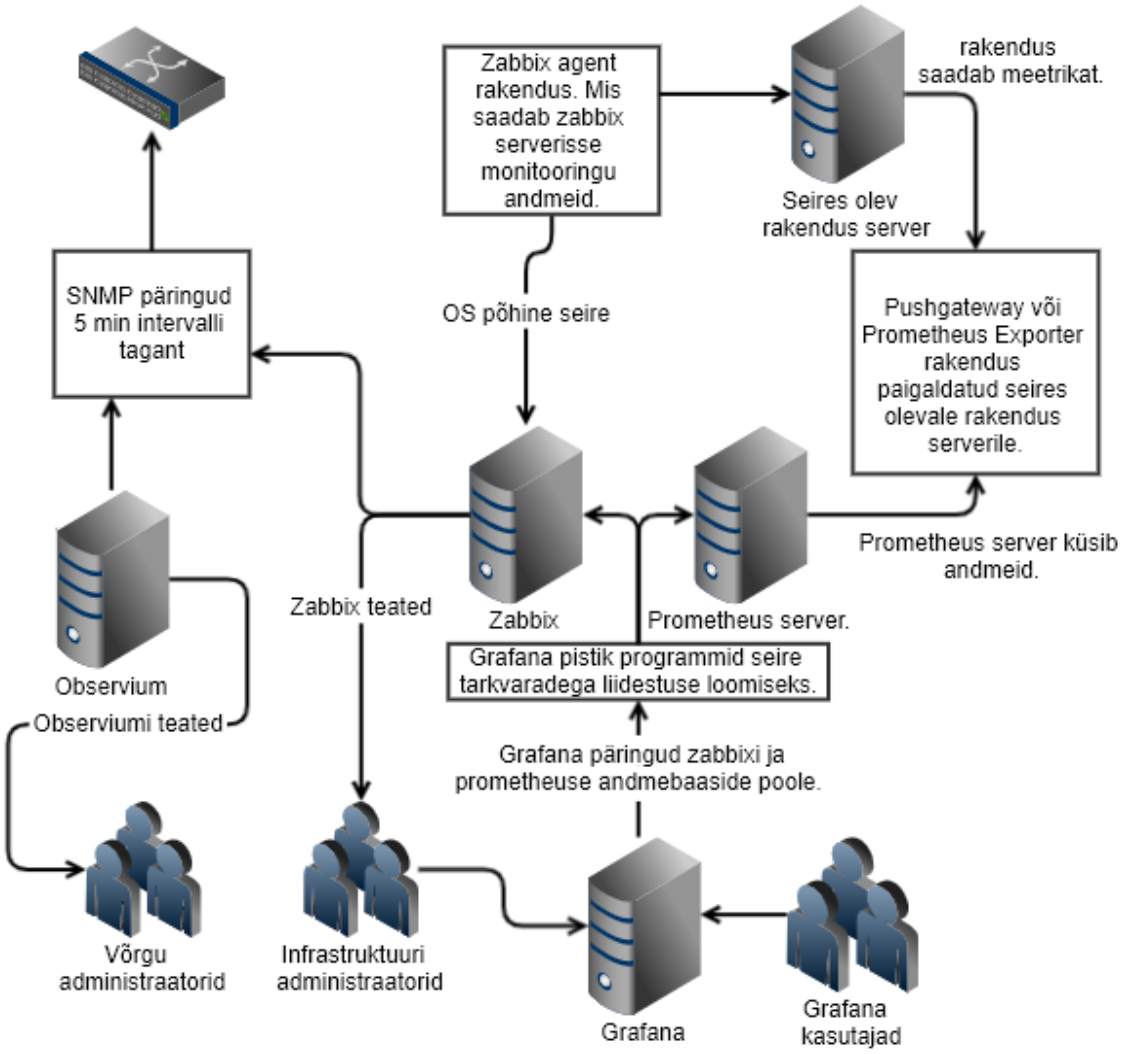
Telegraf-i võib vaadelda kui vahetükina erinevate seiresüsteemide vahel ja tänu sellele on ideaalne viis erinevate teenuste vahelise integratsiooni loomiseks. Näiteks suudab see tarkvara koguda voogtelemeetriat ja suunata selle Prometheus Exporterisse või mõnda andmebaasi. Selle paigaldus on väga lihtne ja nõuab kasutajalt minimaalset seadistamist, kuna kõik pistikprogrammid on sisse ehitatud ja käimapandavad lihtsalt loetava YAML tüüpi seadistusfaili kaudu.

3 TEHIKu seiresüsteemiks vajalike objektide analüüs.

TEHIKu võrguseire vajaduste analüüsiks on vaja vaadelda asutuse võrguseire hetkeolukorda ja kasutusel olevaid seirerakendusi. Antud peatükis analüüsitakse esmalt käsitletava asutuse olemasolevat seiresüsteemi, seejärel voogtelemeetria seire teostamiseks vajalikke komponente.

3.1 Seiresüsteem TEHIK asutuses

Tervise ja Heaolu Infosüsteemide Keskuses on kasutusel mitmed erinevad seiresüsteemid, neist olulisemad on Prometheus, Zabbix ja Observium (Joonis 14).



Joonis 14. TEHIK asutuse tänane seiresüsteem.

Iga seiretarkvara täidab spetsiifilist rolli ja on sisendiks erinevates osakondades, nende kasutus on organisatsiooni tööks kriitilise tähtsusega ja nendest tulenevad sisendid on tähtsaks osaks organisatsiooni töö planeerimisel. Olulisemateks seiretarkvaradeks on järgmised:

- Prometheus – kasutatakse TEHIK asutuse poolt arendatavate teenuste jõudluse jälgimiseks ja on oluline komponent antud rakenduste arendusprotsessis. Selle rakenduse abil jälgitakse kriitiliste süsteemide komponentide tööolekut ja jõudlust.
- Zabbix – on infrastruktuuri ja alusteenuste jälgimise tööriist. Selle sisendiks on serverite jõudlusparameetrid nagu CPU, mälu kasutus, võrguliideste täituvus ja olek jms. Antud tööriist on ka põhiline süsteem vigadele reageerimiseks ja teiste seiresüsteemide töö jälgimiseks.
- Observium – on põhiliselt võrguadministraatorite tööriist, mille abil planeeritakse võrgu ehitustöid, jälgitakse asutuse võrgu olekut ja saadetakse häirete kohta teateid.

Võrguseiret teostatakse mingis osas nii Zabbixi kui ka Observiumi kasutades. Mõlemas süsteemis küll erineval määral ja erinevatel põhjustel, kuid mõlema puhul teostatakse seda kasutades SNMP protokolli või ICMP *ping*-i.

SNMP põhjal teostatav seire on täna väga ebaefektiivne ja ei anna adekvaatset ülevaadet võrguseadmete töö ja võrgu reaalse oleku kohta. Põhjuseks on asjaolu, et SNMP poolt võimaldatav tihedaim andmete kogumise intervall on viis minutit, vastasel juhul muutub seire teostamine võrguseadmete jaoks koormavaks. Selline intervall on kahjuks liiga pikk tänases infosüsteemis, kus enamik rakendusi käivitab oma protsessid väga lühikeseks ajaks. Pika intervalli tagant on võimalik jälgida ainult võrgu üldist olukorda, aga kõik järsud ressursi kasutused jäävad enamasti selliselt nähtamatuks. Ainuke lahendus sellises olukorras on intervalli lühendamine, kuid see ei ole SNMP protokolli kasutades võimalik. Lahenduseks on kasutada uuemaid seire andmete kogumisprotokolle nagu NETCONF või gNMI.

3.2 SNMP analüüs

SNMP-st on saanud tänaseks kõige populaarsem viis võrgu seire teostamiseks, sisuliselt on see olnud ainuke viis võrgu seadmetest seire andmete pärimiseks üle kolmekümne aasta.

Tänapäevases infosüsteemis on SNMP-ga seire teostamine muutunud ebaefektiivseks, suuremahulise andmekoguse juures jääb antud protokoll aeglaseks ja seega ei vasta tänapäevastele ootustele [9].

Järgnevatel põhjustel on TEHIK otsustanud SNMP protokollil põhinevast seirest loobuda ja leida parem viis võrguseire teostamiseks.

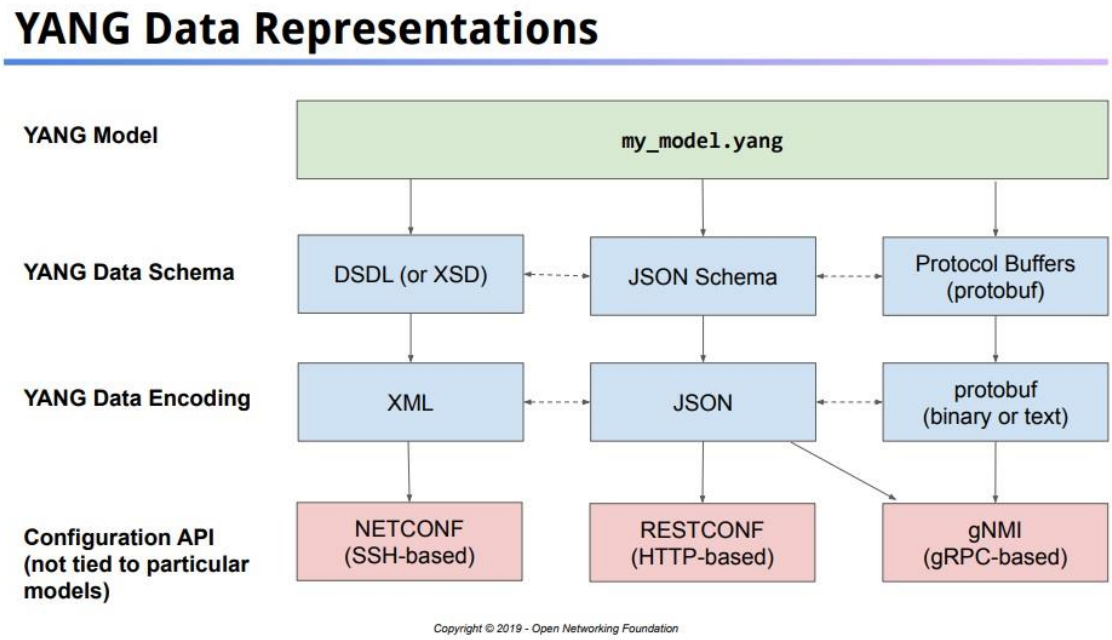
- SNMP pikk intervall seire päringute vahel ei anna adekvaatset ülevaadet süsteemi olekust [9].
- Suuremate päringute peale jääb protokoll aeglaseks [8].
- SNMP OID-d peavad alati olema õiges järjekorras, kuid see ei tähenda, et andmed seadmes tegelikult on sellises järjekorras nagu neid küsitakse. Seetõttu on SNMP päringutele vastamiseks seadmel vaja teha lisa tööd järjestuse korrigeerimiseks. Selline protsess on ressurssi nõudev ja väga suurte andme koguste puhul muutub seadmele koormavaks [8].
- OID väärtuste päringute vastustele ei panda kaasa mõõdistuse ajatemplit, selles saab veenduda SNMP päringute tegemisega. Sellest tulenevalt ei saa kindel olla millal seadme seis tegelikult vastas mõõdetud väärtusele [9].

3.3 Voogtelemeetria protokollide analüüs

Voogtelemeetria jaoks on tänaseks kaks erinevat protokolli: kirjanduse ülevaates mainitud NETCONF, gNMI ja autori poolt välja jäetud RESTCONF, mis on sisuliselt NETCONF üle RESTful rakendusliidese (*Application Programming Interface*, edaspidi API). Selles töös autor RESTCONFI ei käsitle kuid antud kohas väärib see siiski mainimist.

Suurem võrdlus tuleb ette võtta NETCONF-i ja gNMI vahel. gNMI on uuem protokoll, põhineb oluliselt värskematel tehnoloogiatel ja omab mõnevõrra paremaid omadusi kui NETCONF.

Mõlemad protokollid toetavad YANG mudelite-põhist seadmete seadistamist ja voogtelemeetria kogumist. NETCONF teeb seda kasutades XML kodeeringut andmete edastamiseks ja valideerimiseks *Document Schema Definition Languages* raamistikku (edaspidi DSDL) või JSON-it. gNMI eelistatud andmete kodeering tehakse kasutades kas JSON-it või protokolli puhver kodeeringut (Joonis 15) [8].



Joonis 15. NETCONF vs gNMI struktuuri skeem võetud "YANG, OpenConfig, and gNMI Session 2" presentatsioonist [38]

Erinevused kahe protokolli vahel on ka TCP sessiooni alustamise metoodikas, kus eristatakse kahte erinevat sessiooni alustamise viisi [39].

- Dial-in – mis tähendab, et kommunikatsiooni võrguseadme ja andmeid koguva teenuse vahel alustab klient ise ja selle tulemusel alles hakatakse võrguseadmes voogtelemeetriat kliendi poole edastama. Dial-in on dünaamiline ja voogtelemeetria edastuseks vajaminevad seadistused luuakse automaatselt [39].
- Dial-out – puhul alustab TCP sessiooni võrguseade ise ning voogtelemeetriat saadetakse nii pea kui sessioon on alustatud. Selle jaoks on vaja, et andmeid koguv teenus on valmis neid andmeid vastu võtma. Vastupidiselt dial-in meetodile on see staatiline ja kogu seadistus toimub seadme pool [39].

NETCONF toetab ainult Dial-in metoodikat, ja seda YANG-mudeli standardi põhise yang-push ja yang-notify funktsioonidega [40]. gNMI eelpool nimetatud YANG standardeid ei toeta ja selle asemel on arendanud oma funktsionaalsuse mis ei sõltu YANG mudelitest [8].

Tabel 2. NETCONF ja gNMI dial-in ja dial-out funktsionaalsuse võrdlus [8], [24], [39].

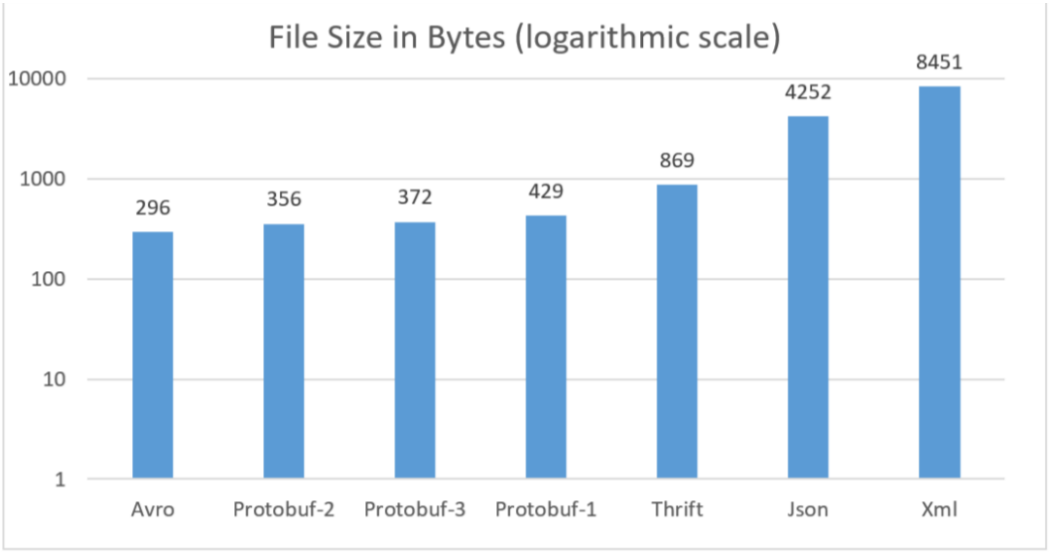
Transport protokoll	NETCONF		gNMI	
	Dial-in	Dial-out	Dial-in	Dial-out
Voogtelemeetria				
	Jah, kasutades yang-push-i või yang-notif-native-t	Ei	Jah, kasutades oma funktsionaalsust.	gNMI spetsifikatsiooni põhjal ei ole, aga on arendamisel, kuid seadmetes võib olla.
Kodeering	XML	-	JSON / protokolli puhver	

Kahe protokolli võrdlemiseks võib kasutada XML-i ja gRPC andmete serialiseemisest tulenevat edastusmahtu ja -kiirust. Kuna mõlemad tehnoloogiad on suhteliselt kaua olnud kasutuses, saab nende vahelise võrdluse jaoks leida mitmeid näiteid. Hea võrdluse on tehtud XML, JSON-i ja protokolli puhvri vahel [41], [42]. Mõlemas artiklis on arvesse võetud serialiseeritava faili suurust, kus testitakse igat kodeeringut suurema ja väiksema andmemahuga.

Võrdluse tegemiseks on arvestatud järgnevate parameetritega:

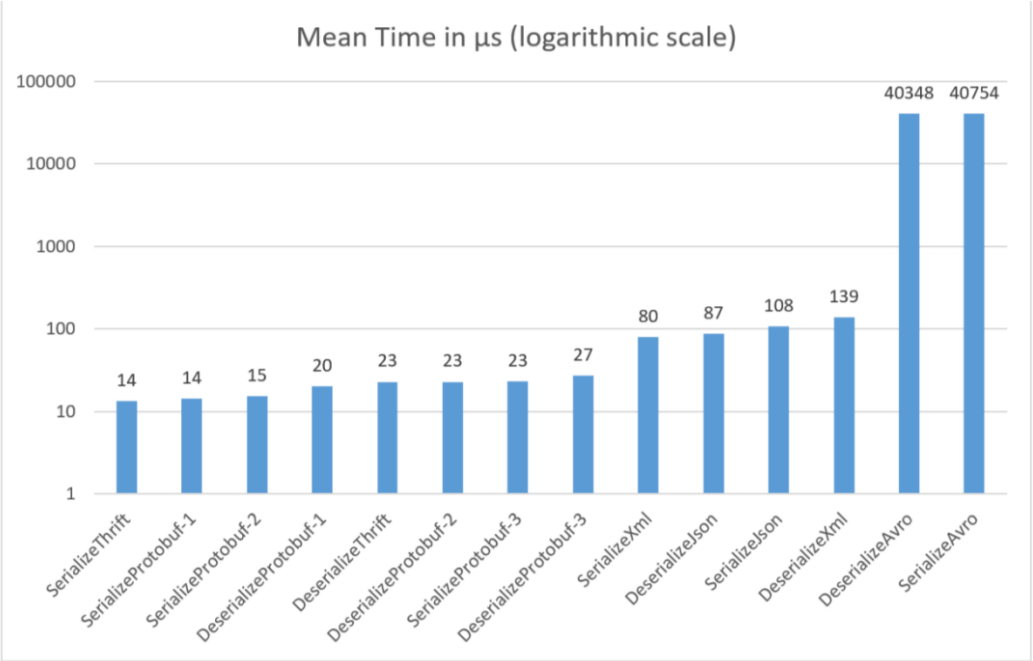
- Serialiseerimisaeg
- Deserialiseerimisaeg
- Serialiseeritud andme maht

Väiksemate andmemahtudega testides tuleb välja, et XML-is serialiseeritud andmete maht on mitmekordselt suurem kui protokolli puhvi puhul, mistahes versiooni protokolli puhvrist kasutati (Joonis 16).



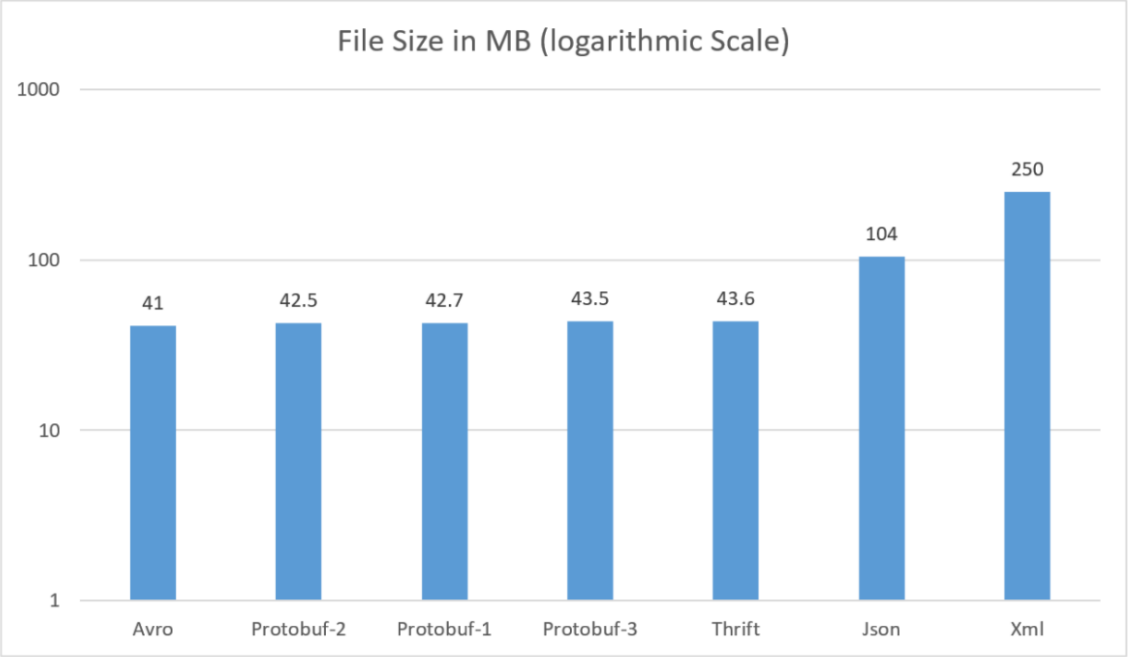
Joonis 16. Kodeeritud andmete suuruse erinevus väikeste andmemahtude serialiseerimise tulemusena, visualiseeritud logaritmiliselt [42].

Antud diagrammil (Joonis 16) on näha, et XML on kõige rohkem andmemahtu kasutav, JSON on kaks korda väiksem. Protokolli puhver on JSON-i ja XML-iga võrreldes kümneid kordi väiksem. Erinevus tuleneb sellest, et nii XML kui JSON kasutavad muutujate nimedeks *string*-tüüpi väärtuseid, kuid protokolli puhver kasutab *string* asemel *integer*-tüüpi väärtuseid oma notatsioonis [42], [43].



Joonis 17. Näide väikese andmemahu serialiseerimisajast, visualiseeritud logaritmiliselt [42].

Andmemahu serialiseerimisaeg on otseselt seotud nii NETCONF-i kui gNMI jõudlusega. Mida kiirem on serialiseerimine, seda kiiremini on seade võimeline saatma serialiseeritud andmed koguvasse teenusesse. Eelnevalt diagrammilt on näha, et XML on jällegi kõige aeglasem ja võrreldes protokolli puhvriga-iga on vahe mitmekümnekordne.



Joonis 18. Kodeeritud andmete suuruse erinevus suurte andmemahtude serialiseerimise tulemusena, visualiseeritud logaritmiliselt [43]

Suuremate andmemahtudega on erinevused vähem märgatavad, kuid siiski mitmekordsed. Arvestades, et voogtelemeetria protokollid saadavad andmeid pideva voona, mitte mitmemegabaidiste hulkadena, on korrektsem võrrelda NETCONF-i ja gNMI jõudlust väiksemate andmemahtude serialiseerimisena.

Muud erinevused NETCONFI ja gNMI vahel on mitte ainult funktsionaalsed vaid ka põhimõttelised, rakenduslikud ja tehnoloogiast tulenevad, mida nende protokollide loomisel kasutati. Põhilised erinevused peale nende, mis eelnevalt juba mainitud, on järgmised:

gNMI eelised NETCONF-i ees:

- gNMI protokolli puhver kodeeringu tõttu on antud protokoll kiirem ja efektiivsem kui NETCONF, kuna kasutab andmete edastamiseks märkimisväärselt vähem võrgu ressurssi [43].
- gNMI jaoks on lihtsam leida valmis tarkvara voogtelemeetria kogumiseks. Tänases olukorras ei ole kirjutatud ühtegi rakendust, mis suudaks NETCONF voogtelemeetriat koguda ilma sellist rakendust ise kirjutamata. On võimalik leida tarkvara mooduleid, mille abil saab lihtsustada antud protokollile rakenduste kirjutamist, kuid valmis tooteid ei ole.
- Erinevalt gNMI-st, kus voogtelemeetria edastamine on osa protokollist, vajab NETCONF laiendit. NETCONF-i lisati see YANG mudelite standardi lisana, mis tõi kaasa *yang-push*-i ja *yang-notify* funktsioonid.
- Tulenevalt gRPC kasutusest on gNMI-d kasutavate rakenduste kirjutamine märksa lihtsam kui teise protokolli jaoks, kuna gRPC dokumentatsioon ja kasutajaskond on märksa suurem.

NETCONF-i eelised gNMI ees:

- Erinevalt NETCONF-ist pole gNMI-il RFC standardit. Põhjuseks on asjaolu, et antud protokolli arendab OpenConfig mitteformaalne töörühm, ja standardi loomine ei ole nende sõnul selle töörühma eesmärk [18].
- Protokolli puhver kodeering on keeruline ja raskesti loetav, lisaks vajab dekodeerimiseks .proto failide olemasolu. Seega on keeruline antud protokolli põhisest kommunikatsioonist aru saada jälgides võrgu liiklust ning kommunikatsioonist tulenevate vigade analüüs raskem [43].

3.4 Voogtelemeetria protokollide vastavus töö nõuetele

Käesolevas töös on võrreldud kahte voogtelemeetria protokolli. Analüüsi käigus selgus, et üks neist on efektiivsem kui teine, kuid mõlemad suudavad täita sama ülesannet. Sellele vaatamata on selle töö alguses välja toodud hulk nõudeid (**Error! Reference source not found.**), millele valitav protokoll peab vastama.

Järgnevates tabelites (Tabel 3 ja Tabel 4) on analüüsitud protokollide vastavust eelpool mainitud nõuetele. Selle tulemusena saab otsustada, milline protokoll on TEHIKu jaoks kõige sobivam.

Tabel 3. gNMI vastavus töö nõuetele.

Nõue	Nõudele vastavus	kirjeldus
1	Jah	Vastab nõudele; iga saadetud sõnum on ajatempliga
2	Jah	Sõltub monitooringu tarkvarast
3	Jah	Võimaldab YANG mudelist küsida kogu seadme andmeid, mis sisaldavad kogu võrguseadme seisukorda, k.a seadme enda ja võrgu liideste kasutus*(t)*
4	Jah	Kogu liiklus seadme ja kliendi vahel on krüpteeritav kasutades TLS protokolli.
5	Jah	Antud protokolli andmete kogumiseks on võimalik kasutada Telegraf või gNMI-gateway tarkvara.
6	Jah	Antud protokoll on seadmete suhtes neutraalne.
7	Jah	Ei sõltu protokollist
8	Jah	Ei sõltu protokollist

Tabel 4. NETCONF vastavus töö nõuetele.

Nõue	Nõudele vastavus	kirjeldus
1	Jah	Vastab nõudele; iga saadetud sõnum on ajatempliga
2	Jah	Sõltub monitooringu tarkvarast
3	Jah	Võimaldab YANG mudelist küsida kogu seadme andmeid, mis sisaldavad kogu võrguseadme seisukorda, k.a seadme enda ja võrgu liideste kasutus*(t)*
4	Jah	Kogu liiklus seadme ja kliendi vahel on krüpteeritav kasutades TLS protokolli
5	Jah	Antud protokollile ei leidnud autor ühtegi valmis kirjutatud tarkvara, mis suudaks koguda voogtelemeetria andmeid

5	Ei	NETCONF on seadmete suhtes neutraalne.
6	Jah	Ei sõltu protokollist
7	Jah	Ei sõltu protokollist

3.5 Analüüsi tulemusel tehtud voogtelemeetria protokolli valik

Eelmistes punktides koostatud voogtelemeetria protokollide analüüsi põhjal võib järeldada, et gNMI on parem valik TEHIK asutuse voogtelemeetria põhise seire teostamiseks, kuna:

- gNMI on efektiivsem tänu oma protokolli puhver kodeeringule. Antud kodeering on küll keerulisem, kuid valmis rakenduste kasutamisel ei ole see asjaolu oluline, kuna rakenduse autor on selle probleemiga tegelenud;
- antud protokolli on lihtsam kasutusele võtta tänu olemasolevatele valmistoodetele nagu Telegraf;
- kõigis teistes osades on NETCONF ja gNMI võrdsed oma funktsionaalsuses.

3.6 Voogtelemeetria andmete kogumine seire rakendustesse

Seitsmes punkt antud töö nõuetes (Tabel 1) on vajadus koguda seire andmed TEHIKus kasutatavatesse seiretarkvaradesse. Tänane olukord on aga selline, et ükski nendest rakendustest voogtelemeetria protokolle ei toeta. Eesmärgini jõudmiseks on autoril vaja leida rakendus, mis looks silla voogtelemeetria andmevoo ja seirerakenduste vahel. Selleks on vaja kõigepealt uurida, milliseid andmeformaate ja protokolle TEHIK seirerakendused kasutavad.

Tabel 5. Seire tarkvarade andmesisestus viiside võrdlus.

Andme sisestus viis	**Prometheus**	**Zabbix**	**Observium**
JSON API	Ei	Jah	Ei
Prometheus Exporter formaat	Jah	Jah	Ei
SNMP	Ei	Jah	Jah
gNMI	Ei	Ei	Ei

Ainus võimalus sisestada voogtelemeetria andmed enamikesse seiretarkvaradesse, on kasutada selleks Prometheusi aegridade andmete formaati, mida oskavad lugeda Zabbix ja Prometheus (Tabel 5). Kahjuks pole Observiumi võimalik muul moel andmeid sisestada kui ainult kasutades selleks SNMP-d.

Täna on võimalik leida kaks erinevat rakendust, mis suudavad voogtelemeetria andmed tõlkida Prometheus *exporter* andmeformaati. Üks nendest on eelpool kirjeldatud Telegraf ja teine on Netflix-i loodud gNMI-gateway.

Telegrafi ja gNMI-*gateway* võrdlus

- Telegraf on valmistoode, n-ö toodang versioonina kasutatav; gNMI-*gateway* seevastu on alles ekperimentaalstaatuses ja arenduse faasis rakendus.
- Mõlemad rakendused on Go keeles kirjutatud ja samaväärselt modulaarsed rakendused.
- Telegrafi kasutamine on tunduvalt lihtsam kui seda on gNMI-gateway.
- gNMI-gateway kasutab sisemiselt andmete modelleerimiseks YANG mudeleid, mis teeb sellest paremini teostatud voogtelemeetria andmete kogumise rakenduse kui Telegrafi gNMI pistikprogrammid.
- Mõlemad rakendused toetavad Prometheus exporter andmete seirialiseerimisformaati. See annab mõlemale samaväärse funktsionaalsuse olema sildühenduseks seiresüsteemi ja voogtelemeetria vahel.
- Telegraf-il on andmete salvestamiseks kasutada InfluxDB, mis tagab kogutud andmete terviklikkuse; gNMI-*gateway*-l andmete salvestamise võimalus puudub.
- Telegraf-i pistikprogrammide arendus on märksa lihtsam võrreldes gNMI-*gateway*-ga, kuna dokumentatsiooni ja näiteid on esimesel tunduvalt rohkem.
- Mõlemad rakendused toetavad TLS protokolli, aga gNMI-*gateway* puhul on selle kasutamine YANG-i standardi tõttu kohustulik.

Antud võrdluse alusel otsustab autor kasutada sildsüsteemina Telegraf-i, peamiselt selle kasutuslihtsuse ja ühilduvuse tõttu, kuid jälgib gNMI-*gateway* arenduskäiku ja tulevikus tuleks kindlasti teha uus analüüs.

4 Planeeritava lahenduse kavand

TEHIKu võrguseire lahenduse analüüsi järelduste põhjal on võimalik kavandada uus seirelahendus, mis katab enamiku töö eesmärkidest ja loob asutuse jaoks uut väärtust.

Voogtelemeetrial põhinev seire süsteem ei ole ainult efektiivsem, vaid on ka täpsem ja detailsem. Selle abil on võimalik tuvastada ka väiksemad anomaaliad asutuse arvutivõrgu töös. Kuid selgus, et ükski asutuses kasutatavatest seire süsteemidest ei toeta voogtelemeetria protokolle ja selle lahenduseks tuleb kasutusele võtta lisarakendus, mille ülesanne on olla sildsüsteem voogtelemeetria andmete kogumise ja seiresüsteemi saatmise vahel.

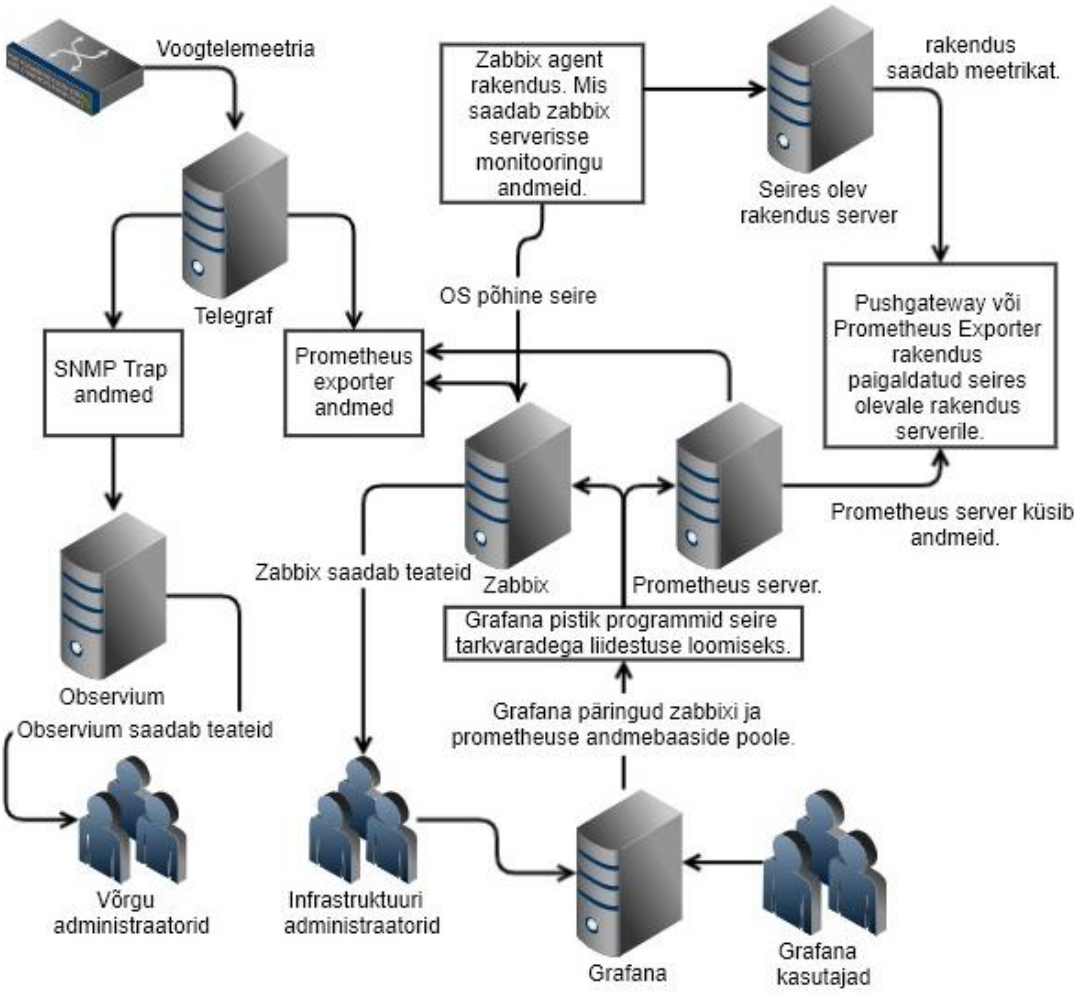
Selliseks sildsüsteemiks on valitud Telegraf, selle kasutuslihtsuse ja valmiduse tõttu. Antud rakendus hakkab koguma kõigi võrguseadmete voogtelemeetria andmeid ja neid edastama seire süsteemidesse.

Seirerakenduste analüüsist selgus, et Prometheus exporter formaati kasutades on võimalik voogtelemeetria andmeid edastada nii Zabbixisse kui Prometheus seirerakendustesse, kuid mitte Observiumi, kuna viimane toetab andmete sisestamise viisina sisuliselt ainult SNMP protokolli, mitte arvestatult skriptide koguga, mida Observiumi autorid kutsuvad agendiks.

Kuid eelneva takistuse lahendamiseks on võimalik Telegrafile kirjutada pistikprogramm, mis suudab edastada SNMP v3 Trap sõnumeid. Sellist pistikprogrammi ei ole väga keeruline arendada ja selle olemasolul saab voogtelemeetriat hakata ka edastama Observiumi.

Kavandatav seirelahendus ei ole suurel määral erinev olemasolevast, põhikomponendid ja nende funktsioon jääb samasuguseks nagu on kirjeldatud kolmandas peatükis.

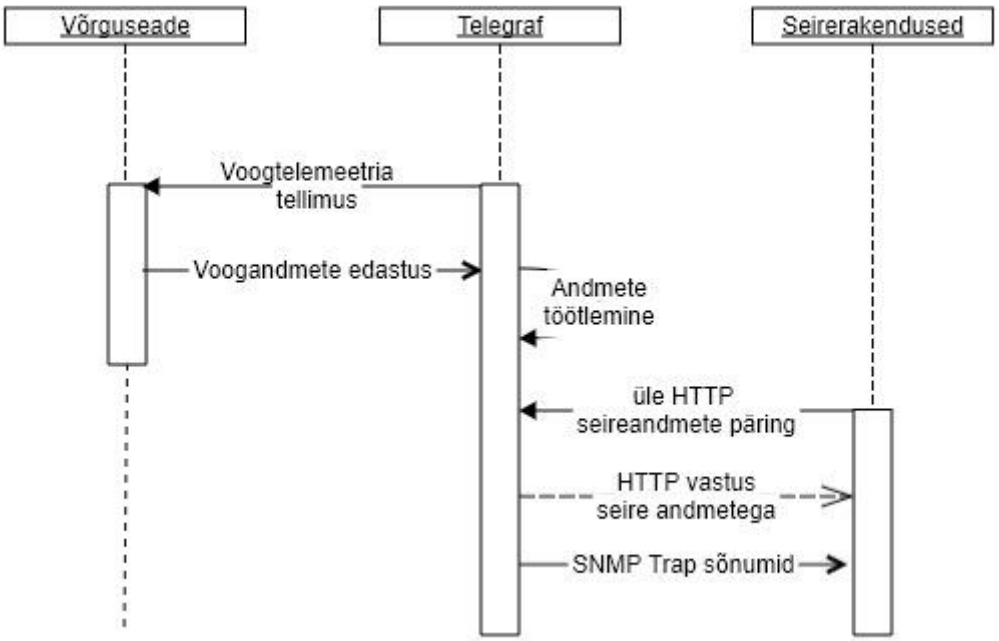
Erinevuseks on SNMP asendamine voogtelemeetria põhise seireandmete kogumisega. Selle saavutamiseks on vaja paigaldada Telegraf iseseisvale serverile. Seejärel seadistada seirerakendused Telegrafi väljundist andmeid koguma. Lisaks on kavandatavas süsteemis võimalik ka Prometheus seirerakendusse võrguseire andmete edastamine, mis varasemalt SNMP põhise seire puhul ei olnud võimalik.



Joonis 19. TEHIK seire süsteem kasutades voogtelemeetriat.

Voogtelemeetria põhise seiresüsteemi komponentide kirjeldus.

- Telegraf ühendab võrguseadmesse ja tekitab selles tellimuse voogandmete kogumiseks.
- Seejärel muudetakse Telegrafi abil voogandmete formaat Prometheus Exporter kujule.
- Zabbix ja Prometheus käivad etteantud intervalli järel Prometheus Exporter liidesest andmeid kogumas.
- SNMP Trap liidese arenduse järel hakkab Telegraf voogtelemeetria andmeid saatma Observiumi.
- Seirerakendused seadistatakse vastavalt antud rakenduse juhistele andmeid töötlema ja muutustele reageerima.



Joonis 20. Voogtelemeetria andmete töötlus protsess.

5 Kontseptsiooni tõestus labori keskkonnas

Enne asutuses kasutusele võtmist on vajalik kavandi põhine süsteem realiseerida labori keskkonnas, et tõestada selle praktilisus ja rakendatavus. Selleks on autor ette valmistanud test keskkonna, kuhu on võimalik paigaldada kõik eelpool nimetatud rakendused sarnastes oludes, mida on oodata ka reaalses infrastruktuuris.

Keskkond, kuhu paigaldatakse virtualiseeritult vajalikud Linux serverid põhineb Citrix *Hypervisor* platvormil, millest lähemalt võib lugeda selle toote koduleheküljelt [44]. Antud platvorm võimaldab paigaldada vajaliku hulga virtualiseeritud Linux servereid, kuhu omakorda saab paigaldada kõik vajalikud rakendused.

Labori keskkonnas testib autor lahendust Zabbixil ja Prometheusel, kuna töö kirjutamise ajal SNMP Trap sõnumite saatmiseks vajalikku Telegraf pistik programmi veel arendatud ei ole.

Labori keskkonnaks vajalikud füüsilised komponendid on järgnevad.

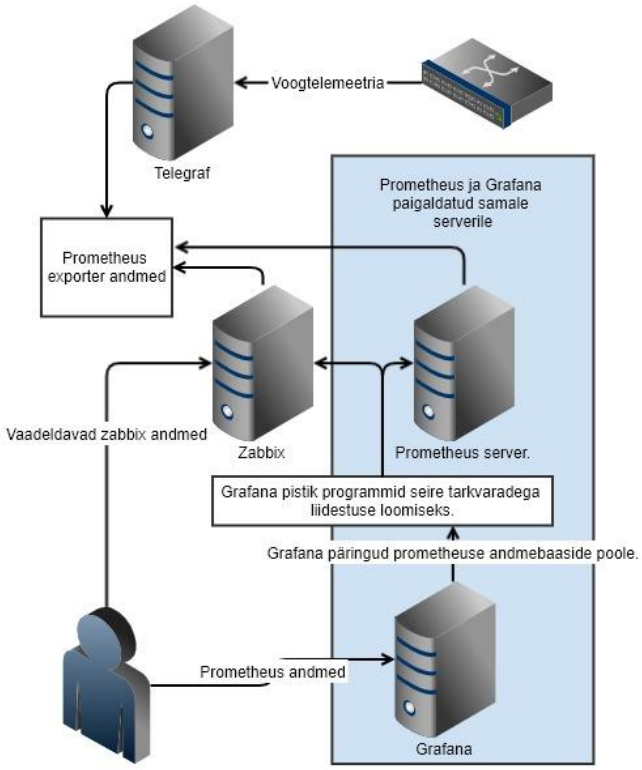
- Juniper tootja switch, millel on kõik tööks vajalikud funktsionaalsused ning mis on asutuse võrguadministraatori poolt ette valmistatud voogtelemeetria andemete edastamiseks. Võrguseadme täpne mudel siinkohal ei ole oluline, kuna tootja kasutab kõigis oma seadmetes samasugust operatsioonisüsteemi ja protokolle. Oluline on ainult märkida, et operatsioonisüsteemi versioon antud seadmes on 20.2R1.10, mis on ühtlasi töö kirjutamise ajal kõige värskem versioon.
- IBM Thinkstation server, millel on piisaval hulgal füüsilist ressurssi mitme virtualiseeritud operatsioonisüsteemi käivitamiseks ja töös hoidmiseks.

Virtualiseeritud on kolm Ubuntu Linux operatsioonisüsteemi versiooniga 20.04, kõik keskkonnad on identsed.

Antud operatsioonisüsteem sai valitud selle kasutuslihtsuse ja autori eelneva kogemuse tõttu. Samasuguse tulemuse saab saavutada ka teiste Linux põhiste operatsioonisüsteemidega.

1. Keskkond millele on paigaldatud Telegraf ja mis on seadistatud koguma Juniper switchist voogtelemeetriat.

2. Prometheuse ja Grafana rakenduste jaoks paigaldatud keskkond. Esimene kogub andmeid Telegrafist kasutades selleks Prometheus Exporter formaati ja Grafana on seadistatud kuvama kogutud andmeid. Reaalses keskkonnas oleks need eraldi serveritele paigaldatud, aga antud test keskkonnas pole selleks vajadust.
3. Zabbix keskkond, mis samuti hakkab koguma andmeid Telegrafist kasutades selleks eelpool nimetatud formaati, selline funktsionaalsus on antud tarkvara uuematesse versioonidesse sisse ehitatud.



Joonis 21. Labori keskkonna joonis.

5.1 Telegrafi paigaldus ja seadistamine

Telegraf rakenduse paigaldamiseks oli kõige lihtsam viis järgida toote koduleheküljel olevat juhendit [45]. Antud juhend on lihtsasti loetav ja järgitav. Pakkudes samm-sammult juhiseid, mille tulemusel on paigaldatud toote kõige värskem versioon, mis töö kirjutamise hetkel oli 1.17.

Telegrafi paigalduse järel tuli seadistada antud rakendus võrguseadmest andmeid koguma, ka selle jaoks on koduleheküljel põhjalik juhend saadaval. Näide autori labori seadistusest on järgnev.

```
[agent]
  flush_interval = "10s"

[[inputs.gnmi]]
  addresses = ["192.168.1.9:50051"]
  username = ""
  password = ""

  encoding = "proto"
  redial = "10s"

  [[inputs.gnmi.subscription]]
    path = "/components/component[name='Routing Engine0']"
    subscription_mode = "sample"
    sample_interval = "2s"

  [[inputs.gnmi.subscription]]
    path = "/interfaces/interface/state"
    subscription_mode = "sample"
    sample_interval = "2s"
```

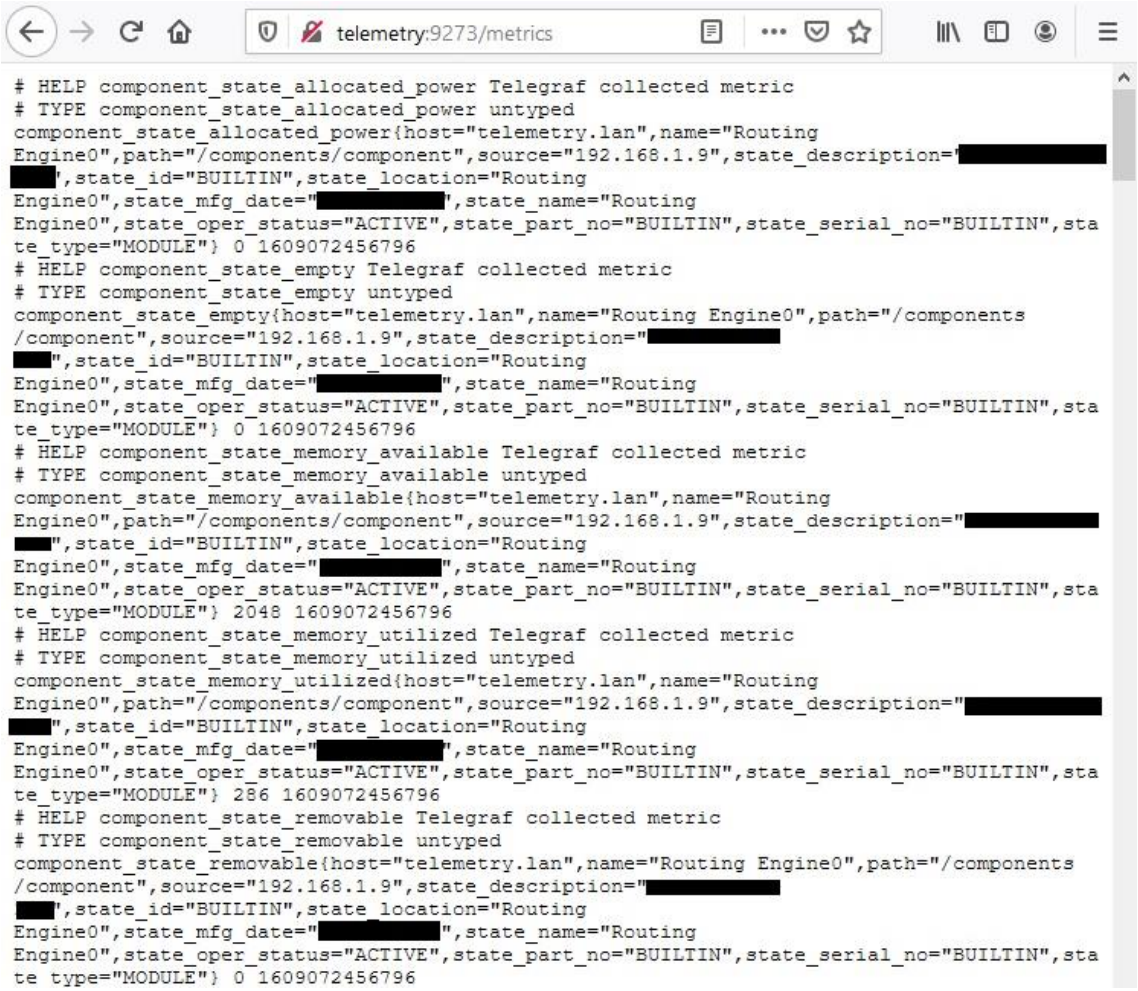
Joonis 22. Telegraf labori sisendi seadistus.

```
[[processors.enum]]

  fielddrop = ["properties/property/state/configurable"]
  [[processors.enum.mapping]]
    ## Name of the field to map. Globs accepted.
    field = "properties/property/state/value"

    ## Table of mappings
    [processors.enum.mapping.value_mappings]
      "Online" = 1
      "Offline" = 0
      "REV 15" = 15
      "0x1:power cycle/failure" = 1
      "0x2:watchdog" = 2
      "0x6:thermal shutdown" = 6
      "Online Master" = 0
      "Master" = 1
      "EX2300-C-12P" = 0

[[processors.converter]]
  [processors.converter.fields]
    integer = ["properties/property/state/value"]

  [processors.converter.tags]
    measurement = ["/components/component/properties/property/name"]

[[outputs.prometheus_client]]
  listen = ":9273"
  metric_version = 2
  collectors_exclude = ["gocollector", "process"]
  export_timestamp = true
  expiration_interval = "30s"
```

Joonis 23. Telegraf labori väljundi seadistus.

Seadistuse järel ja teenuse käivitamisel on Telegraf paigaldatud ja valmis jagama voogtelemeetria andmeid üle Prometheus Exporter-i, määratud pordil, antud serveri aadressil.

Oluline on märkida, et kuigi Telegraf võimaldab voogtelemeetria andmete krüpteerimist kasutades TLS protokolli ja Prometheus Exporteri andmete krüpteerimiseks on võimalik kasutada SSL-i siis lihtsuse mõttes, kumbagi meedet labori näites kasutatud ei ole.



Joonis 24. Näide Telegraf-i väljundist, mida kuvab Prometheus Exporter.

5.2 Zabbix seiretarkvara paigaldus ja seadistamine

Sarnaselt Telegrafi rakendusele on ka Zabbixi jaoks selge ja põhjalik paigaldusjuhend koduleheküljelt saadaval [28]. Seda kasutades paigaldas autor eelpool mainitud Linux keskkonda nimetatud rakenduse kõige värskema versiooni, mis oli töö kirjutamise hetkel 5.2.

Paigaldusjuhendit samm-sammult järgides on tulemuseks paigaldatud Zabbixi seirerakendus, mis on valmis Telegraf-ist andmete päringute tegemiseks.

Edasiseks tegevuseks järgis autor toote koduleheküljelt juhiseid, mis selgitavad Prometheus exporter-ist seire andmete päringute tegemist. Kuna tegemist on mahuka

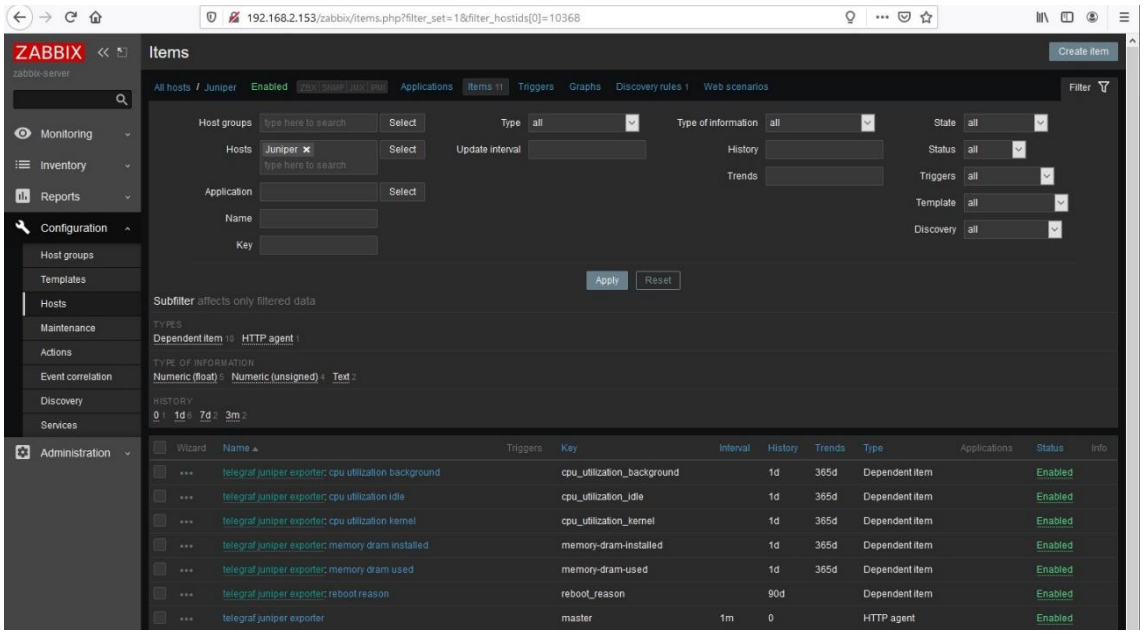
juhisega, siis selle protsessi põhjalik kirjeldamine antud töös ei ole mõistlik ja lugeja saab sellega ise tutvuda antud leheküljel [46].

Juhise järgimise tulemusena on Zabbix serverrakendus alustanud päringute tegemist Telegrafist ja sealt saadud andmed on valmis seiresüsteemis kasutamiseks.

Oluline erinevus eelpool mainitud juhendist on seire objektide seadistus. Labori loomise käigus selgus, et kohati on telegrafist saadetavad andmed duplikaatidega. See tuleneb asjaolust, et Prometheus formaat ei toeta *string* tüüpi seire andmete väärtuste edastust.

Lahenduseks on Zabbixis olemas funktsionaalsus, mis suudab duplikaadid eemaldada, mille tulemusel arvestatakse ainult kõige värskema tulemuse väärtusega.

Eelpool nimetatud juhise põhjal sai loodud Zabbix serveri konfiguratsiooni Juniper nimeline seade, selle alla nn *master* objekt, mis kogub Telegrafi poolt ettevalmistatud andmeid Zabbixi andmebaasi edasiseks töötluseks.



Joonis 25. Juniper seadmest tulevate andmete seadistus Zabbixis.

Eelpool olevas näites (Joonis 25) on näha Juniper nimelist seadet Zabbix seiresüsteemis ja mainitud *master* objekti, millele on pandud nimeks „telegraaf juniper exporter“. Lisaks on näha juba hulka töödeldud seire objekte, mis võimaldavad seadme olekut jälgida.

Erinevalt eelpool mainitud Prometheus integratsiooni loomise juhendist tuli autoril teha seire objektid vastavalt antud olukorrale ja andmete omadustele, mida saab kõige paremini kirjeldada näite abil.

Näiteks võime valida Juniper seadme CPU temperatuuri ja antud seadme parameetri seadistamist Zabbixisse. Selle töö protsessi alustuseks tuli Telegrafi väljundist leida huvipakkuv parameeter ja vaadelda selle omadusi (Joonis 26).

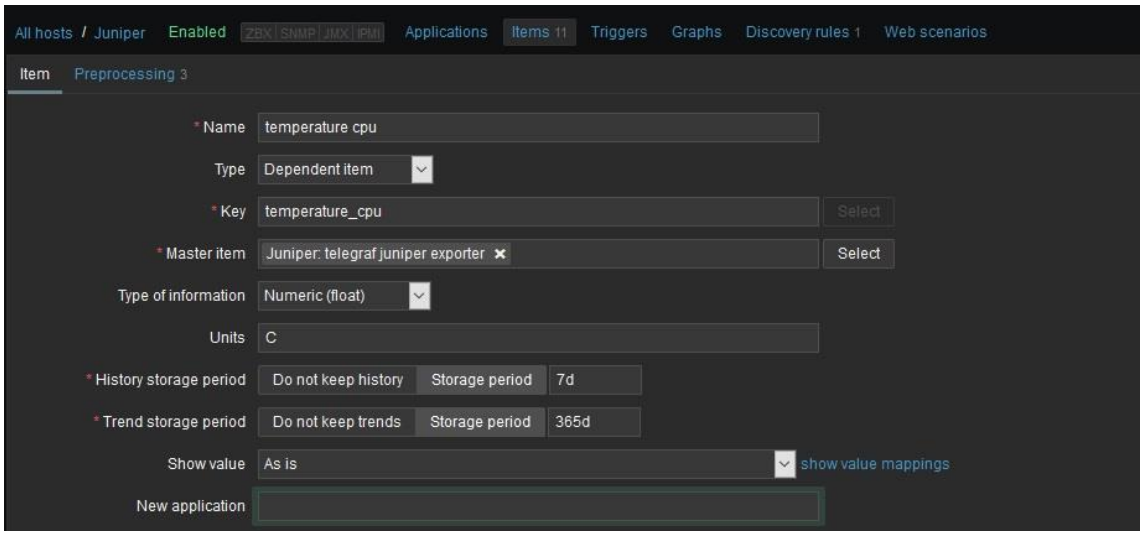
```
component_state_temperature_instant{
        host="telemetry.lan",
        name="Routing engine0",
        path="/components/component",
        source="192.168.1.9",
        state_location="Routing Engine0",
        state_mfg_date="09-18-2018",
        state_name="Routing Engine0",
        state_oper_status="ACTIVE",
        state_part_no="BUILTIN",
        state_serial_no="BUILTIN",
        state_type="MODULE"
} 52
```

Joonis 26. Prometheus Exporterist väljavõte seadme CPU temperatuuri kohta.

Eelnevas näites on olulisteks osadeks järgnevad asjad:

- Mõõdistuse nimi (*metric name*), milleks antud näites on „components_component_properties_property_state_configurable“
- Objekti väärtust hoidev silt (*label*), mis asub loogeliste sulgude vahel ja mille nimetus on antud näites „component_properties_property_state_value“
- Objekti nime hoidev silt „components_component_properties_property_name“

Kui objekti parameetrid olid välja otsitud, sai neid kasutades seadistada Zabbix seire keskkond. Selleks tuli luua uus seire objekt, mida Zabbixis kutsutakse *Item*. Antud objekti omadused tuli seadistada järgmiselt (Joonis 27).

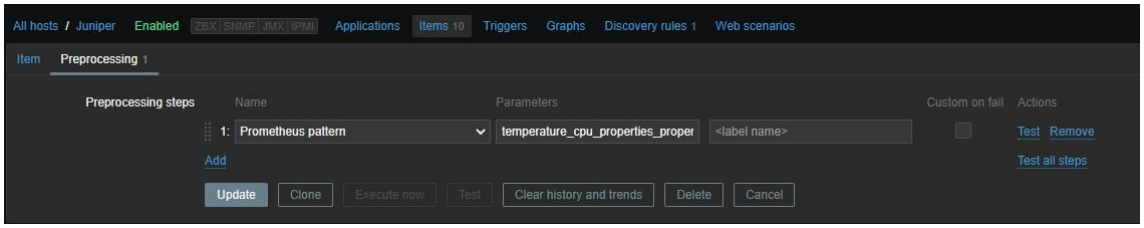


Joonis 27. Juniper seadme CPU temperatuuri seireobjekti seadistus Zabbixis.

Olulisemateks parameetriteks on selles seadistuses vaja märkida järgnevad parameetrid:

- *Name* – vabas vormis seire objekti nimi
- *Type* – mille väärtus on d*ependent item* ehk teisest objektist sõltuv objekt.
- *Key* – mis on samuti vabas vormis kasutaja määrata nimetus objektile kuid ei tohi sisaldada tühikuid.
- *Master item* – objekti nimi millest antud objekt sõltub.
- *Type of information* – määrab kogutavate seire andmete tüübi, antud näiteks on selleks *float* tüüpi reaalarvud.
- *Units* – Mis on märgistatud C-ga ka on lühend Celsiusest.

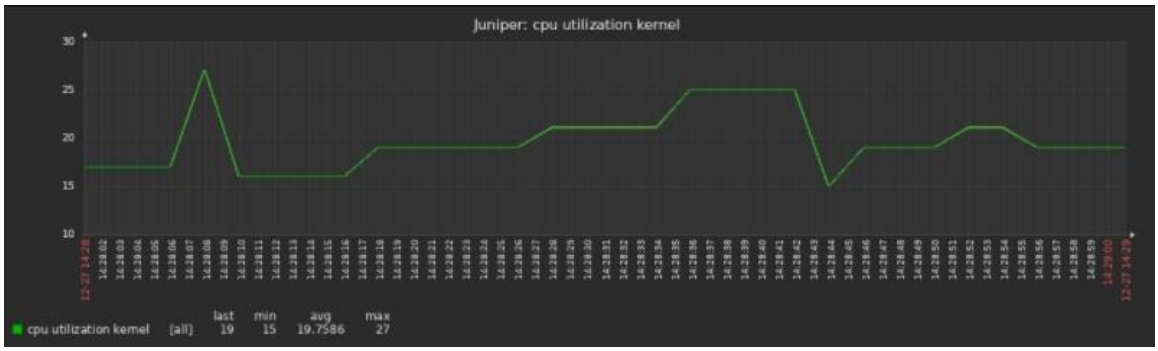
Seireobjekti seadistuse järel tuli defineerida andmete töötlusprotsess. Selle käigus töötleb Zabbix, Prometheus Exporter formaadist tulevad andmed sellisele kujule, mida on võimalik salvestada Zabbixi andmebaasi.



Joonis 28. Zabbix seireobjekti töötlus

Antud andmetöötlust tuleb teha iga seireobjekti jaoks. Nagu näha Joonis 28, tuli selleks määrata ainult mõõdu nimi, milleks on selles näites „*component_state_temperature_instant*“, Zabbix seejärel kasutab mõõdu väärtust seireobjekti väärtusena.

Samasugust protsessi kasutades on seadistatud laboris hulk seadme parameetreid, huvitavama näitena võib välja tuua CPU kasutuse, täpsemalt kerneli poolt kasutatava ressursi seiregraafiku viimase minuti vältel. Tänu voogtelemeetriale on muutusi graafikul näha iga kahesekundise intervalli järel (Joonis 29). See on oluliselt täpsem kui SNMP viieminutiline intervall.



Joonis 29. Zabbix seire süsteemis olev seadme CPU kasutuse ühe minutilise intervalliga.

Antud näidetes on tõestatud, et voogtelemeetria andmete põhjal on võimalik teostada võrguseadmete seiret kasutades selleks Zabbixit.

Tööprotsess ei ole väga keeruline, ega vaja palju tarkvara komponente ja seadistamist. Kogu Zabbixi poolse töö saab teha kasutades selleks Zabbixi sisseehitatud funktsionaalsust.

5.3 Prometheus serveri ja Grafana seadistamine

Prometheus serveri paigalduseks tuli järgida toote koduleheküljelt leitavat paigaldusjuhendit [47]. Juhendit järgides tuli seadistada Prometheus andmeid koguma Telegrafi väljundist. Selle saavutamiseks tuli kirjutada järgmine konfiguratsiooni fail (Joonis 30), mis sisaldas minimaalset vajalikku seadistust rakenduse käivitamiseks ja andmete kogumiseks.

```
# my global config
global:
  scrape_interval:     2s
scrape_configs:
    - job_name: 'telemetry.lan'
      static_configs:
      - targets: ['telemetry:9273']
```

Joonis 30. Minimaalne Prometheus serveri seadistus.

Selle seadistuse põhiselt käib Prometheus värskeid andmeid küsimas iga kahesekundilise intervalliga. Täpsemalt saab iga parameetri kohta lugeda toote paigaldusjuhendist [47].

Säärase seadistuse tulemusena on Prometheus server paigaldatud ja minnes serveri veebi aadressile pordil :9090 on võimalik kogutavaid andmeid lähemalt näha.

Prometheuse andmete visualiseerimiseks tuleb paigaldada labori keskkonda Grafana. See paigaldati samale serverile nagu Prometheus, kasutades Grafana koduleheküljel olevat paigaldusjuhendit [35]. Paigalduse järel tuli seadistada Grafana kasutaja liidese abil Prometheus pistikprogrammi parameetrid, mille järel sai hakata tegema päringuid Prometheuse andmebaasi poole. Selle tulemusena sai hakata looma Grafana vaateid (Joonis 31), mis kuvavad kasutajale seadme töö oleku staatust kahesekundilise intervalliga.

Joonis 31. Juniper seadme vaade Grafanas.

Nagu (Joonis 31) on näha, jõuavad voogtelemeetria andmed Grafana vaadetesse ja sellega on tõestatud, et kavandatud lahendus vastab ootustele.

5.4 Järeldused ja edasine töö

Töö alguses sai püstitatud hulk nõudmisi ja ootusi voogtelemeetria põhise seire kohta (Tänastes IT-süsteemides on tavaline, et käivitatakse väga lühikeseks ajaks suure koormusega rakendusi, mis kasutavad arvestatavat võrgu ressurssi, aga töötavad kõigest mõnikümmend sekundit. Klassikalise võrguseire praktika järgi küsitakse võrguseadmetelt nende töö parameetreid iga mõneminutilise intervalli tagant, mis võib autori kogemuste kohaselt olla ühe- kuni kümneminutilise vahega, mis tähendab, et eelpool nimetatud rakenduste töö mõju avastamine on selliselt väga ebatõenäoline.

Lahenduseks on tänaseks välja töötatud voogtelemeetria, mis erinevalt klassikalisest lähenemisest, kus mingi intervalli tagant seadmest päritakse selle staatust, saab nüüd võrguseade ise oma staatuse andmeid pideva andmevoona seiresüsteemi saata.

Tulenevalt käsitletava asutuse infrastruktuuri nõuetest, seirevajadustest ja tänastest kitsaskohtadest, on koostatud lühike nimekiri ootuseid ja nõudmisi voogtelemeetria-põhise seirekeskkonna loomisele (Tabel 1).

Tabel 1) eesmärgil rakendada see TEHIK keskkonnas võrguseire teostamiseks.

Siinkohal on aeg üle vaadata, kuidas voogtelemeetria eelpool nimetatud ootustele vastab ja teha järeldused selle sobivuse kohta TEHIKu keskkonda oodatud eesmärki täitma.

Tabel 6. Labori vastavus töö nõuetele.

Nõue	Saavutatud	Kommentaar
1.	Jah	Voogtelemeetria andmed on kõik märgitud ajatempliga, erinevalt SNMP-st kus seire andmetel ajatemplit küljes ei ole, lisaks sellele on gNMI protokolliga andmete tihedus kordades suurem.
2.	Jah	Olukorras kus andmete saatmine on peatunud tuleb seire tarkvaral ise sellele reageerida. Õnneks on selline funktsionaalsus enamikes seiretarkvarades olemas.
3.	Jah	Voogtelemeetria protokollide abil saab küsida kõiki ja seadistada enamikke seadme parameetreid.

4.	Jah	Labori keskkonnas seda küll ei testitud, kuid rakenduste ja seadmete dokumentatsiooni ja gNMI spetsifikatsiooni kohaselt on kogu andmevahetus krüpteeritav.
5.	Jah	Telegraf mis sai valitud voogtelemeetria andmete kogumiseks võimaldab gNMI protokollist andmeid koguda ilma mingi lisa arenduseta.
6	Jah	gNMI ja OpenConfig YANG mudelid mida Telegraf-iga laboris kasutati on täielikult seadmete suhtes neutraalsed. Sama lahendust on võimalik kasutada olenemata seadme tootjast.
7	Jah, mööndustega	Enamike seire rakenduste sisendiks on võimalik voogtelemeetria andmeid kasutada ilma mingi arendustööta, välja arvatud Observium-i jaoks, tulevikku jääb otsustada kas Observium on edasi kasutatav või tuleb siiski ette võtta arendustöö Telegrafi ja Observiumi vahelise ühenduse loomiseks.
8.	Jah	Kõik rakendused mis olid vajalikud labori ehitamiseks töötasid Linux operatsiooni süsteemil.

Analüüsi ja labori realisatsiooni tulemusena saab edasiseks tööks planeerida TEHIK asutusse antud seire lahenduse rakendamise. Üldises vaates hakkab reaalne süsteem välja nägema väga sarnane labori keskkonnaga, erinevuseks saab olema krüpteeritud andmevahetus ja seadmete arv ning sellest tulenevalt mõningad muudatused Telegrafi seadistuses.

Edasiseks tööks jääb Observiumi jaoks voogtelemeetria ja SNMP vahelise Telegrafi pistikprogrammi loomine, mille abil oleks võimalik ka viimasesse seirerakendusse voogtelemeetria andmed edastada.

Kuigi voogtelemeetrial põhinev seire lahendus eksisteerib töö kirjutamise ajal ainult labori keskkonnas, on alustatud TEHIK asutuses toodangu keskkonna ettevalmistust uue seiresüsteemi arenduseks. Arvestades töö mahu ja keerukusega, loodetakse sellega valmis saada käesoleva aasta jooksul.

Kokkuvõte

Üle kolmekümne aasta on kasutatud võrguseire teostamiseks SNMP protokolli, see tehnoloogia on olnud interneti ajastu üks olulisemaid alustalasid aidates silma peal hoida kõigil võrguseadmetel, mis tagavad tänapäevase eluviisi.

Kuid tehnoloogia arengu, suuremate ootuste ja internetti ühendatud seadmete hulga kiire kasvuga ei suuda SNMP enam sammu pidada. Selle asjaolu tõttu on suurte ettevõtete nagu Netflix ja Google võrguinsenerid kokku tulnud ja välja töötanud uuemad protokollid seire ja halduse teostamiseks, mis kasutavad tänapäevaseid tehnoloogiaid ja värskeid ideid.

Erinevalt SNMP-st suudavad voogtelemeetria protokollid saata võrgu seadmetest seire andmeid sekundiliste intervallidega, andmed on krüpteeritud ja seadmete ressurssi ei koormata lõputute päringutega. Sellise andmetihedusega on märgata ka kõige väiksemaid muutusi infosüsteemi keskkonnas.

Kuid uute protokollide tekkimisega ei ole veel kõik probleemid lahendatud. Nagu sellest tööst on võimalik lugeda, ei toeta enamik seiresüsteeme voogtelemeetria kogumist ilma lisatarkvara vahenduseta, vanemad võrguseadmed ei oma vajalikku funktsionaalsust ja uute protokollide kasutamiseks vajaminev õppimiskõver on suhteliselt suur.

Uus tehnoloogia on alles lapsekingades, kuid omab suurt potentsiaali ja järgneva aastakümnendi jooksul muudab võrguhalduse ja seire maastikku tundmatuseni, parandades nähtavust ja vähendades reageerimisaega infosüsteemides toimuvatele muutustele.

Selle töö eesmärk oli rakendada voogtelemeetrial põhinev võrguseire Tervise ja Heaolu Infosüsteemide Keskuses ja selle lõppeesmärgini on pikk tee veel ees. Kindlasti on eesmärk saavutatav, kuna suuremaid takistusi töös esitatud labori lahenduses ei tekkinud.

Täna võib öelda, et SNMP aeg on möödas, selle panus on antud ja see on meid hästi teeninud, kuid nüüd tuleb liikuda uues suunas ja julgelt kasutusele võtta värskemad tehnoloogiad. Voogtelemeetria protokollid, olles küll värsked ja arengufaasis, on piisavalt kaugele arenenud, et kasutada oma igapäevases võrguhalduse ja seiretöös.

Kasutatud kirjandus

[1] M. Douglas R. ja K. Schmidt, „1. Introduction to SNMP and Network Management - Essential SNMP, 2nd Edition“. https://learning.oreilly.com/library/view/essential-snmp-2nd/0596008406/ch01.html (vaadatud dets 18, 2020).

[2] V. G. Cerf, „IAB recommendations for the development of Internet network management standards“. https://tools.ietf.org/html/rfc1052 (vaadatud dets 18, 2020).

[3] M. T. Rose ja K. McCloghrie, „Structure and identification of management information for TCP/IP-based internets“. https://tools.ietf.org/html/rfc1155 (vaadatud dets 18, 2020).

[4] M. T. Rose ja K. McCloghrie, „Concise MIB definitions“. https://tools.ietf.org/html/rfc1212 (vaadatud dets 18, 2020).

[5] J. Davin, J. D. Case, M. Fedor, ja M. L. Schoffstall, „Simple Network Management Protocol (SNMP)“. https://tools.ietf.org/html/rfc1157 (vaadatud dets 18, 2020).

[6] M. Douglas R. ja K. Schmidt, „Essential SNMP“, *059600020L*. https://docstore.mik.ua/orelly/networking_2ndEd/snmp/ch02_03.htm (vaadatud dets 18, 2020).

[7] „SNMP MIB Explorer - Juniper Networks“. https://apps.juniper.net/mib-explorer/navigate.jsp (vaadatud dets 31, 2020).

[8] B. Claise, J. Clarke, ja J. Lindblad, „Network Programmability with YANG: The Structure of Network Automation with YANG, NETCONF, RESTCONF, and gNMI, First Edition“. https://learning.oreilly.com/library/view/network-programmability-with/9780135180471/ (vaadatud dets 21, 2020).

[9] J. Edwards, „Streaming telemetry challenges SNMP in large, complex networks“, *Network World*, sept 29, 2020. https://www.networkworld.com/article/3575837/streaming-telemetry-gains-interest-as-snmp-reliance-fades.html (vaadatud dets 14, 2020).

[10] M. Bjorklund, „YANG - A Data Modeling Language for the Network Configuration Protocol (NETCONF)“. https://tools.ietf.org/html/rfc6020 (vaadatud dets 19, 2020).

[11] B. and B. C. Lengyel, „YANG Instance Data File Format“, mai 2018. https://tools.ietf.org/id/draft-ietf-netmod-yang-instance-file-format-00.html (vaadatud dets 19, 2020).

[12] „Learn YANG - Full Tutorial for Beginners“. https://ultraconfig.com.au/blog/learn-yang-full-tutorial-for-beginners/ (vaadatud dets 19, 2020).

[13] J. Schoenwaelder, „Overview of the 2002 IAB Network Management Workshop“. https://tools.ietf.org/html/rfc3535 (vaadatud dets 19, 2020).

[14] „FAQ: YANG Labs“. https://community.juniper.net/browse/blogs/blogviewer?blogkey=5b7d279d-b8ba-47c0-8ece-3ba0d417ba18 (vaadatud dets 19, 2020).

[15] M. Wasserman <mrw@painless-security.com>, „Using the NETCONF Protocol over Secure Shell (SSH)“. https://tools.ietf.org/html/rfc6242#section-3.1 (vaadatud dets 19, 2020).

[16] R. Enns, M. Bjorklund, ja J. Schoenwaelder, „Network Configuration Protocol (NETCONF)“. https://tools.ietf.org/html/rfc6241 (vaadatud dets 19, 2020).

[17] J. Edelman, „OpenConfig, Data Models, and APIs“, *Jason Edelman's Blog*. http://jedelman.com/home/openconfig-data-models-and-apis/ (vaadatud dets 19, 2020).
[18] „OpenConfig - FAQ“. https://www.openconfig.net/docs/faq/ (vaadatud dets 19, 2020).
[19] „OpenConfig - Home“. https://www.openconfig.net/ (vaadatud jaan 03, 2021).
[20] „FAQ“, *gRPC*. https://grpc.io/docs/what-is-grpc/faq/ (vaadatud dets 20, 2020).
[21] „Protocol Buffers“, *Google Developers*. https://developers.google.com/protocol-buffers (vaadatud dets 20, 2020).
[22] S. Popić, D. Pezer, B. Mrazovac, ja N. Teslic, „Performance evaluation of using Protocol Buffers in the Internet of Things communication“, okt 2016, lk 261–265, doi: 10.1109/SST.2016.7765670.
[23] D. Rao, „Telemetry in Action: NETCONF and gNMI with a Custom-Built Collector!“, *Cisco Blogs*, sept 24, 2020. https://blogs.cisco.com/datacenter/telemetry-in-action-netconf-and-gnmi-with-a-custom-built-collector (vaadatud dets 20, 2020).
[24] R. Shakir, C. Lebsack, A. Shaikh, S. Benoit, M. Albano, ja R. Dodin, „OpenConfig Reference“, *GitHub*. https://github.com/openconfig/reference/blob/master/rpc/gnmi/gnmi-specification.md (vaadatud jaan 02, 2021).
[25] „Supported languages“, *gRPC*. https://grpc.io/docs/languages/ (vaadatud dets 31, 2020).
[26] „Observium“. https://docs.observium.org/ (vaadatud dets 20, 2020).
[27] „Zabbix :: The Enterprise-Class Open Source Network Monitoring Solution“. https://www.zabbix.com/ (vaadatud dets 31, 2020).
[28] „Zabbix Manual [Zabbix Documentation 5.0]“. https://www.zabbix.com/documentation/5.0/manual (vaadatud dets 20, 2020).
[29] R. Rihards Olups, A. Andrea Dalle Vacche, ja P. Patrik Uytterhoeven, „Zabbix: Enterprise Network Monitoring Made Easy“. https://learning.oreilly.com/library/view/zabbix-enterprise-network/9781787129047/ (vaadatud dets 20, 2020).
[30] „Global Dashboard“. https://www.zabbix.com/global_dashboard (vaadatud dets 20, 2020).
[31] „Zabbix Agent“. https://www.zabbix.com/zabbix_agent (vaadatud dets 20, 2020).
[32] „Overview | Prometheus“. https://prometheus.io/docs/introduction/overview/ (vaadatud dets 20, 2020).
[33] „Ephemeral Containers“, *Kubernetes*. https://kubernetes.io/docs/concepts/workloads/pods/ephemeral-containers/ (vaadatud dets 21, 2020).
[34] R. Lingappa, „Ephemeral Jobs Monitoring Using Prometheus PushGateway“, *Medium*, juuli 14, 2020. https://itnext.io/ephemeral-jobs-monitoring-using-prometheus-pushgateway-917b33486564 (vaadatud dets 21, 2020).
[35] „Getting started“, *Grafana Labs*. https://grafana.com/docs/grafana/latest/getting-started/ (vaadatud dets 21, 2020).
[36] „Prometheus - Demo Dashboard - Grafana“. https://play.grafana.org/d/000000029/prometheus-demo-dashboard?orgId=1&refresh=5m (vaadatud dets 21, 2020).
[37] „Telegraf Open Source Server Agent“, *InfluxData*. https://www.influxdata.com/time-series-platform/telegraf/ (vaadatud dets 21, 2020).

[38] B. O'Connor, „Next-Gen SDN Tutorial - Session 2: YANG, OpenConfig, and gNMI". Open Networking Foundation, Vaadatud: sept 26, 2019. [Online]. Available at: https://opennetworking.org/wp-content/uploads/2019/10/NG-SDN-Tutorial-Session-2.pdf.
[39] Cisco, „Programmability Configuration Guide, Cisco IOS XE Gibraltar 16.10.x - Model-Driven Telemetry [Cisco IOS XE Gibraltar 16.10.1]", *Cisco*. https://www.cisco.com/c/en/us/td/docs/ios-xml/ios/prog/configuration/1610/b_1610_programmability_cg/model_driven_telemetry.html (vaadatud dets 23, 2020).
[40] E. Voit, A. Clemm, E. Nilsen-Nygaard, A. Prieto, ja A. Tripathy, „Dynamic Subscription to YANG Events and Datastores over NETCONF". https://tools.ietf.org/html/rfc8640 (vaadatud dets 19, 2020).
[41] M. Novak, „Serialization Performance comparison(XML,Binary,JSON,P...)", *M@X on DEV*, märts 25, 2014. https://maxondev.com/serialization-performance-comparison-c-net-formats-frameworks-xmldatacontractserializer-xmlserializer-binaryformatter-json-newtonsoft-servicestack-text/ (vaadatud dets 23, 2020).
[42] A. Khan ja F. Jardon, „Data Serialization Comparison", *Criteo Labs*, mai 17, 2017. https://labs.criteo.com/2017/05/serialization/ (vaadatud dets 23, 2020).
[43] S. Cadora, „Streaming Telemetry with Google Protocol Buffers", *Cisco Blogs*, mai 11, 2016. https://blogs.cisco.com/sp/streaming-telemetry-with-google-protocol-buffers (vaadatud dets 17, 2020).
[44] „Citrix Hypervisor | Open Source Server Virtualization". https://xenserver.org/ (vaadatud dets 25, 2020).
[45] „Install Telegraf | Telegraf 1.17 Documentation". https://docs.influxdata.com/telegraf/v1.17/introduction/installation/ (vaadatud dets 26, 2020).
[46] „Zabbix 4.2 — Prometheus Integration – Zabbix Blog". https://blog.zabbix.com/zabbix-4-2-prometheus-integration/7558/ (vaadatud dets 26, 2020).
[47] „Installation | Prometheus". https://prometheus.io/docs/prometheus/latest/installation/ (vaadatud dets 26, 2020).

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

Mina, Gerd Kukemilk

1. Annan Tallinna Tehnikaülikoolile tasuta loa (lihtlitsentsi) enda loodud teose „Voogtelemeetria-põhine võrguseadmete seire Tervise ja Heaolu Infosüsteemi Keskuse näitel“, mille juhendaja on Priit Raspel
   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.

01.01.2021

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.