TALLINNA TEHNIKAÜLIKOOL

Raadio- ja sidetehnika instituut

Kood: IRT70LT

Andmeside kursuse võrguhalduse laboritöö Simple Network Management Protocol baasil

Simple Network Management Protocol lab assignment for telecommunication course

Märt Erik

Töö on tehtud telekommunikatsiooni õppetooli juures

Juhendaja: Indrek Rokk

Kaasjuhendaja: Avo Ots

Kaitsmine toimub raadio- ja sidetehnika instituudi kaitsmiskomisjonis

Autor taotleb tehnikateaduse magistri kraadi

Esitatud: 31. jaanuar 2011

Kaitsmine: 8. veebruar 2011

Tallinn 2011

Referaat

Käesolev magistritöö „Andmeside kursuse võrguhalduse laboritöö Simple Network Management Protocol baasil“ käsitleb telekommunikatsiooni magistriõppe tudengite jaoks loodud laboritööd, mis hõlmab võrguhalduse teostamist protokolliga Simple Network Management Protocol ja tutvustab infrastruktuuri seireks mõeldud tarkvara Nagios.

Käesoleva magistritöö eesmärgiks on koostada Andmeside kursust läbivate tudengite jaoks laboritöö võrguhalduse kohta tuginedes maailma ülikoolides tehtavatele Simple Network Management Protocoli harjutustele ja autori kogemustele. Antud töös vaadeldakse Simple Network Management Protocoli toimimist ja struktuuri, tutvustatakse viite võrguseire tarkvara, analüüsitakse viite maailma erinevates ülikoolides kasutatavat võrguhalduse laborit, koostatakse võrguhalduse laboritöö ja antakse ideid laboritöö edasiarenduseks. Magistritöö esimest poolt, kus antakse ülevaade protokollist ja võrguseire programmidest, kasutatakse labori teoreetilises osas ja teist poolt, kus analüüsitakse maailma ülikoolide laboreid ning koostatakse tudengite jaoks harjutused, kasutatakse labori praktilises osas.

Magistritöö koosneb 109 leheküljest, sisaldab 17 joonist ning on kirjutatud eesti keeles.

Magistritöö võtmesõnad on haldusinfobaas, objekti identifikaator, Simple Network Management Protocol, protsess, Nagios.

Abstract

Master thesis „Simple Network Management Protocol lab assignment for telecommunication course“ covers lab assignment which is created for telecommunication master students and includes network management with Simple Network Management Protocol. Lab assignment also introduces Nagios software which is used for monitoring infrastructure.

This master thesis gives overview about Simple Network Management Protocol operations and structure, introduces five network monitoring software, analyzes five network management lab assignments used in different universities all over the world, composes network management lab assignment and suggests ideas for developing lab in future. Present master thesis purpose is to compose lab assignment about network management for students who take course Andmeside (Data Communications). This lab assignment is based on Simple Network Management Protocol exercises done in world’s different universities and author’s experiences. First half of thesis, where is overview about protocol and network monitoring programs, is used in lab theoretical part and second half of thesis, where is analysis of labs used in different universities all over the world and is composed exercises for students, is used in lab practical part.

Master thesis consists of 109 pages, includes 17 figures and is written in Estonian.

Master thesis keywords are management information base, object identifier, Simple Network Management Protocol, process, Nagios.

Eessõna

Võrguhaldus on alati info- ja kommunikatsioonitehnoloogia tugevas huviorbiidis, sest tõrgeteta toimiv infrastruktuur ja probleemidest koheselt teavitav infosüsteem on suureks abiks ettevõtete normaalsel ja jätkusuutlikul tegevusel. Siinkohal ei oma tähtsust kas ettevõte on suur korporatsioon või väikefirma, milles on paarkümmend arvutit. Võrguhaldus on oluline valdkond telekommunikatsiooniga tihedalt seotud inseneridele ja seega leidis magistritöö autor, et telekommunikatsiooni tudengid peaksid saama praktilise kogemuse võrguhalduse vallas. Infrastruktuuri haldamisel on põhiliseks abivahendiks protokoll Simple Network Management Protocol ja kuna magistritöö autoril on tööülesannete täitmisel kokkupuuteid võrguhaldusega ning nimetatud protokolliga ja huvi antud valdkonna vastu, siis otsustas autor luua laboritöö, mis annaks tudengitele praktilise kokkupuute võrguhaldusega. Lisaks leidsid ka Tallinna Tehnikaülikooli infotehnoloogia teaduskonna raadio- ja sidetehnika instituudi telekommunikatsiooni õppetooli õppejõud, et võrguhalduse valdkond on oluline ning seda tuleks tudengitele tutvustada ja seetõttu on tekkinud ka ülikoolil reaalne vajadus võrguhalduse laboritöö järgi.

Käesolevas magistritöös tutvustab autor esmalt protokolli Simple Network Management Protocol, millel laboritöö põhineb. Samuti antakse ülevaade mõningatest programmidest, mida kasutatakse võrguseire jaoks. Töö oluliseks osaks on maailma erinevates ülikoolides kasutatavate võrguhalduse laborite harjutuste analüüs uurimaks kas nendest saab rakendada ka midagi loodavas laboritöös. Magistritöö autor võtab küll ideid erinevatest ülikoolidest, kuid töö põhiline osa ehk labori koostamine on tehtud töö koostaja isiklikel kogemustel, sealjuures arvestades, et tudengile tuleb võimalikult lihtsalt ja selgelt võrguhalduse valdkonda tutvustada.

Antud magistritöö eesmärgiks on koostada võrguhalduse laboritöö telekommunikatsiooni magistriõppe tudengitele. Eesmärk saavutatakse kasutades laboris magistritöö praktilist osa kui ka teoreetilist osa. Teoreetiline osa põhjal antakse tudengitele ülevaade

protokollist Simple Network Management Protocol ja võrguseire vahenditest, et tudeng oleks kursis võrguhalduse põhiteadmistega. Laboritöö koostatakse Andmeside kursuse jaoks.

Autor tänab oma juhendajat Indrek Rokki ja kaasjuhendajat Avo Otsa heade soovituste ja nõuannete eest.

Sisukord

Jooniste loetelu

Kasutatud lühendid ja terminid

Lühendite ja terminite selgitused põhinevad allikal [1].

AppleTalk - kohtvõrgu arhitektuur ja protokoll, mis on ehitatud kõigisse Apple Macintoshi arvutitesse ja laserprinteritesse ning mis võimaldab omavahel ühendada Macintoshi arvuteid ja printereid, aga ka IBM PC tüüpi arvuteid, kui need on varustatud spetsiaalse Appletalki riist- ja tarkvaraga.

ASN.1 - *Abstract Syntax Notation One* - sides ja arvutivõrkudes kasutatav standard võrgus edastatava sõnumi (rakendusandmeüksuse) kirjeldamiseks, kodeerimiseks, edastamiseks ja dekodeerimiseks.

GPL - *General Public License* - litsents, mis garanteerib kasutajatele vabaduse tarkvara levitada ja modifitseerida.

IANA - *The Internet Assigned Numbers Authority* - üksus, mis korraldab ülemaailmselt IP aadresside jaotamist, domeeninimede süsteemi juurtsooni haldamist ja hoolitseb muude IP protokolliga seotud ülesannete lahendamise eest.

IP - *Internet Protocol* - internetiprotokoll. Protokoll ehk reeglistik, mida järgitakse andmepakettide saatmisel ühelt arvutilt teisele üle interneti.

IPX - *Internetwork Packet Exchange* – võrkudevaheline paketivahetus.Võrguprotokoll firmalt Novell, mis ühendab omavahel Novell NetWare kliente ja servereid kasutavaid võrke.

OSI - *Open Systems Interconnection architecture* - avatud süsteemide sidumise arhitektuur. ISO ja ITU-T koostöös 1977.a. valminud andmesideprotokollide

kontseptuaalne mudel, kus OSI seitsmekihilise arhitektuuriga baasmudel annab loogilise struktuuri konkreetsetele andmesidevõrkude standarditele.

PDU - *Protocol Data Unit* - protokolliandmeüksus. Andmetest ja juhtinfost koosnev andmeüksuste pakett, mida kaks võrgusõlme antud protokolli alusel omavahel vahetavad.

Protokollipinu - *stack* - tarkvaraliselt realiseeritud kihiline protokollistik, mis tagab võrgufunktsioonide täitmise, mille puhul iga vahepealne protokollikiht kasutab allpool asuvat kihti ja teenindab ülalpool asuvat kihti.

RFC - *Request For Comments* - kommentaarinõue. Interneti kohta käivate dokumentide seeria.

SSH - *Secure Socket Shell, Secure SHell* - turvaline kest, turvakest. UNIXi põhine käsuliides ja protokoll, mis võimaldab turvalist sisselogimist kaugarvutisse.

TCP - *Transmission Control Protocol* - edastusohje protokoll. Transpordikihi protokoll, mis lisab internetiprotokollile töökindla sideühenduse ja andmevoo reguleerimise.

Telnet - Interneti kaug-sisselogimise standardprotokoll.

UDP - *User Datagram Protocol* - kasutajadatagrammi protokoll. Tarnspordikihi protokoll, mis pakub teenust andmete vahetamisel internetiprotokolli kasutavasse võrku ühendatud arvutite vahel, kuid erinevalt edastusohje protokollist ei tegele UDP sõnumi jagamisega pakettideks (datagrammideks) ja nende õiges järjekorras kokkuühendamisega vastuvõtupoolel.

Sissejuhatus

Info- ja kommunikatsioonitehnoloogia omab tänapäeval väga suurt tähtsust- tõenäoliselt ei ole valdkonda, kus infothenoloogiliste vahendite olulisus tagaplaanile jäetakse. Praktiliselt kõik ettevõtted kasutavad oma äriprotsessides võrguseadmeid, servereid ja arvuteid ning erinevaid tarkvararakendusi, et firma tööd efektiivselt juhtida ja võimalikult suurt kasumit teenida. Samuti kasutavad ja kindlasti vajavad infotehnoloogia vahendeid riigiasutused, et teha elanikkonnale avaliku informatsiooni kättesaamine võimalikult lihtsaks ja tagada rahva jaoks oluliste (transport, energia, haridus jne) kui ka äärmiselt kriitiliste (kiirabi, tuletõrje, päästeteenistus jne) teenuste kättesaadavus. Seetõttu on nii ettevõtetele kui riigiasutustele vajalik, et nende infrastruktuur ja pakutavad infotehnoloogiateenused oleksid alati töökorras ja kätte saadavad. Vastasel korral võib ettevõtte kasum kannatada või maine kahjustada saada ning riigil teenuste elanikeni toimetamine häiritud olla või halvemal juhul hättasattunule abi saatmata jääda. Infrastruktuuri komponentide 100%-line töösoleku tagamine on praktiliselt võimatu, sest paratamatult juhtub õnnetusi või tekib rikkeid. Samas on aga oluline probleemsest allikast ja probleemi põhjusest kiiresti teada saada, et viga kõrvaldada ja tagada sellega kiire infrastruktuuri töö taastumine ja teenused. Siinkohal on lahenduseks võrguhaldus, mille käigus jälgitakse infrastruktuuris olevaid seadmeid ja tarkvararakendusi. Rikete esinemisel saadakse praktiliselt koheselt teada probleemi olemusest ning allikast ja teavitatakse sellest kindlaid isikuid, kes parandavad vea. Võrguhalduse ja monitooringu abil saab tagada infrastruktuuri võimalikult efektiivse toimimise ja kõrge töösoleku protsendi.

Magistritöö eesmärk on välja mõelda ja reaalselt valmis teha võrguhalduse seisukohast väga olulise komponendi- Simple Network Management Protocoli (edaspidi SNMP)-laboritöö. Labor on mõeldud kursuse Andmeside jaoks. Nimetatud õppeaines on juba olemas võrguhalduse praktikum, kuid see on küllaltki ühekülgne, tutvustades põhiliselt ainult võrguliikluse jälgimist, ja liialt vähe protokolli ennast. Samas on SNMP kogu võrguhalduse alus, mille tõttu otsustas töö autor luua labori, mis kõigepealt annab

protokollist ja selle toimimisest ülevaate ja seejärel laseb õpilastel ise praktilise kogemuse abil SNMP-ga toimetada. Sealjuures on labori käigus soov näidata üliõpilastele, et SNMP võimalused on väga laiad ja ei piirdu ainult võrguliikluse jälgimisega. Pärast labori läbimist ja oluliste teadmiste omandamist SNMP-st oleks kasulik ja vajalik üliõpilastele tutvustada teisigi võrguhalduse aspekte vaadates võrguhalduse teemale laiemalt (ehk kuidas hallata ja toime tulla nende asjadega, millega SNMP ei saa hakkama või mida saab olemasolevale monitooringusüsteemile juurde integreerida, et SNMP abil saadavad seire tulemused annaks olulist lisaväärtust). Praktikumitunni koostamise juures on mõtteid võetud ka maailma ülikoolides tehtavatest SNMP laboritest.

Autori eesmärk on oma tööga anda reaalne sisend telekommunikatsiooni õppesuuna üliõpilastele praktilise kokkupuute saamiseks SNMP-ga, kuid eelkõige laiemas mõttes võrguhaldusega. Lisaks protokolli tundmaõppimisele annab autor labori käigus ka kogemuse maailmas ühe populaarseima võrguseireprogrammi Nagios [60] kasutamiseks, et üliõpilane saaks soovi korral lahendust iseseisvalt kasutada. Kuna internetis kasvav kuritegevus informatsiooni varastamisel ja identiteedi võltsimisel kasvab päev- päevalt järjest suuremaks [63], siis on labori üks eesmärke kindlasti rõhutada turvalisust SNMP kasutamisel. Võrguhalduse puhul liigub nii sisevõrkudes kui ka avalikus internetis ettevõtete jaoks olulist infrastruktuuri haldusinformatsiooni, mis pahatahtlikesse kätesse sattudes võivad tekitada suurt kahju. Labori käigus näidatakse kuidas SNMP liiklus saab turvaliselt nii sisevõrkudes kui ka avalikus võrgus toimuda.

Magistritöö seletuskiri koosneb viiest osast. Esimeses osas on detailne ülevaade SNMP-st, selle toimimisest, struktuurist ja turvalisusest. Teises osas tutvustab autor viite populaarset võrguseirevahendit, annab ülevaate õppejõudude poolt esitatud nõuetest labori jaoks ning põhjendab laboris tutvustatava seirevahendi valikut. Kolmandas osas kirjeldatakse mõningad maailma erinevates ülikoolides kasutatavaid SNMP laboreid ning selgitatakse mida ja miks on kaasatud ka antud töö laborisse. Neljas osa on konkreetse laboritöö kirjeldus, mis autori poolt kokku pandud Andmeside kursuse jaoks. Viiendas osas on toonud töö koostaja mõningaid soovitusi, kas siis antud magistritöö labori edaspidiseks täiendamiseks või lisalabori tegemiseks võrguhalduse kohta. Soovituste

mõte on muuta labor veelgi praktilisemaks ja luua vajadusel juurde teine praktikum, et laiendada infrastaruktuuri seire võimalusi kohtades, kus SNMP-st enam ei piisa.

Kuna eestikeelset kirjandust on antud teema kohta väga vähe, siis üritab koostaja korrektselt tõlkida inglisekeelset teksti ja termineid. Mõnes kohas on sulgudes eestikeelse teksti taga kirjas ka ingliskeelne tähendus, et lugeja jaoks oleks paremini mõistetav, millega tegu. Samas jätab autor siiski mõned sõnad inglisekeelseks kuna neile polegi eesti keeles sobivat vastet ning arusaadavus jääb oluliselt paremaks.

1. Võrguhalduse protokoll SNMP

SNMP on protokoll haldamaks seadmeid IP võrkudes. Mitmed erinevat tüüpi seadmed kui ka programmid toetavad SNMP-d, sealhulgas marsruuterid, kommutaatorid, serverid, tööjaamad, printerid, puhvertoiteallikad (UPS), Unix ja Windows operatsioonisüsteemid jne. SNMP kasutamise võimalused varieeruvad täiesti tavapärastest haldustegevustest väga harvaesinevate soovideni: lihtne ja igapäevani asi on jälgida näiteks marsruuterite, kommutaatorite, serverite riistvara korrasolekut, kuid SNMP abil on võimalik ka seadmeid hallata ja tekkivate probleemide korral käivitada automaatseid tegevusi probleemide eemaldamiseks. Informatsioon, mida on võimalik kontrollida, varieerub suhteliselt lihtsatest ja standardsetest elementidest- näiteks võrguliikluse maht, mis läbib võrguliidest- kuni väga spetsiifiliste riistvara objektideni- näiteks kindla tootja marsruuteri protsessori temperatuur. [2, lk ix]

SNMP tuumikuks on tegevuste kogumik (ja loomulikult informatsioon, mida nende tegevuste käigus kogutakse), mis annab võrgu haldajale võimaluse pärida ja sealjuures ka muuta SNMP-d toetavate seadmete objektide olekut (objektide mõiste on lahti seletatud punktis „1.1.3 Haldusinfostruktuur“). Iga seade, mille peal töötab tarkvara, mis lubab pärida SNMP informatsiooni, on ka antud protokolli abil hallatav. See ei hõlma ainult füüsilisi seadmeid (ehk riistavara andmeid) vaid ka tarkvara nagu näiteks veebiserverid ja andmebaasid. [2, lk 1]

SNMP laiahaardelisusest on siinkohal väga hea näide John Romkey loodud *Internet Toaster* ehk interneti röster. Ise kirjeldab mees uudse leiutise loomise tagamaid järgnevalt: “Ajal kui ma olin ettevõttes Epilogue, lõime me 1990. aastal interneti röstri Interopi (iga-aastane äritehnoloogia üritus koos konverentside ja uute infotehnoloogiavõimaluste tutvustamisega [7]) ajaks. Minu eesmärk oli panna inimesed mõtlema SNMP-st kui mitte lihtsalt muutujate pärimise võimalusest vaid rakenduste kontrollimisest ehk siis laiemast vaatenurgast. Meil on röster, mida sai juhtida SNMP-ga.

Kui sa panid saia röstrisse ja andsid SNMP-ga käsu, siis hakkas röster saia küpsetama. [6]“

Hetkel on võrguhalduse protokollist SNMP olemas kolm versiooni: SNMP versioon 1, SNMP versioon 2 ja SNMP versioon 3. Versioon 1 on protokolli esimene versioon, mis on defineeritud RFC-dega 1155 ja 1157. Versioon 2 on versiooni 1 edasiarendus ja seda põhiliselt haldusinfostruktuuri (haldusinfostruktuuri on selgitatud punktis „1.1.3 Haldusinfostruktuur“) ja protsesside (SNMP protsesse selgitab paragrahv „1.2 Protsessid“) osas. Versioon 3 kirjeldab ära SNMP turvalise versiooni. SNMP erinevate versioonide turvalisusest on räägitud paragrahvis „1.3 Turvalisus“. [59]

1.1 SNMP võrguhalduse struktuur

Protokolli SNMP toimimisest ja protsessidest aru saamiseks ning SNMP abil informatsiooni kogumise ja seadmise mõistmiseks selgitatakse kõigepealt lahti põhilised protokolliga seotud komponendid. Samuti on kirjeldatud, mis on erinevate komponentide ülesanded, kuidas need komponendid omavahel arvutivõrkudes suhtlevad ja missugune on nende komponentide struktuur.

1.1.1 Üksused

SNMP protokollistikus on kahte tüüpi üksused, mille vahel toimub võrguhaldusteadete vahetamine: haldaja ja agent. Haldaja on server, mille peal töötab haldusülesandeid täitev tarkvara infrastruktuuri seadmete ja tarkvara seireks. Haldaja kohta öeldakse üldiselt *Network Management Station* ehk võrguhaldusjaam (edaspidi kasutatakse lühendit NMS). Agent on tarkvara, mis töötab jälgitavate võrguseadmete peal. Agenttarkvara võib olla täiesti eraldi programm (näiteks Unixis töötav deemon) või juba operatsioonisüsteemi tootja poolt liidetud tarkvara (näiteks Cisco marsruuterite peal olevasse operatsioonisüsteemi IOS on agent vaikimisi lisatud). Tänasel päeval on enamikesse võrguseadmetesse SNMP agent juba sisse ehitatud [2, lk 3].

Joonisel 1 (asub leheküljel 18) on üksused toodud selliselt, et vasakul poolel olev arvuti on võrguhaldusjaam ja paremal pool on jälgitavad seadmed, mille peal töötab agent.

NMS on vastutav päringute (*poll*) ja *trap*'ide vastuvõtmise eest agentidelt, mis asuvad võrgus. Pärimine võrguhalduse kontekstis tähendab, et agendilt (marsruuter, Unixi server jne) küsitakse mingisugust informatsiooni seadme seisundi kohta. Seda informatsiooni saab hiljem kasutada, et kindlaks teha kas on toimunud mõni sündmus. *Trap*'id on agendi viis öelda NMS-ile, et midagi on seadmega juhtunud. *Trap*'id saadetakse asünkroonselt ehk mitte vastuseks NMS poolt tehtud päringule. NMS on seejärel vastutav edasiste tegevuste eest *trap*'ilt saadud info põhjal. Näiteks kui võrguliides enam ei funktsioneeri, saadab marsruuteri SNMP agent NMS-ile *trap* teate. Selle peale viib NMS läbi mingi tegevuse, näiteks saadab administraatorile telefoni peale sõnumi, et midagi on juhtunud. Siinkohal on oluline aru saada, et päringud kui ka *trap*'ide saatmine võivad toimuda samaaegselt. Selle kohta ei ole mingisuguseid piiranguid, millal haldusjaam võib agendilt informatsiooni pärida või millal agent ise saadab *trap*'i. [2, lk 3]

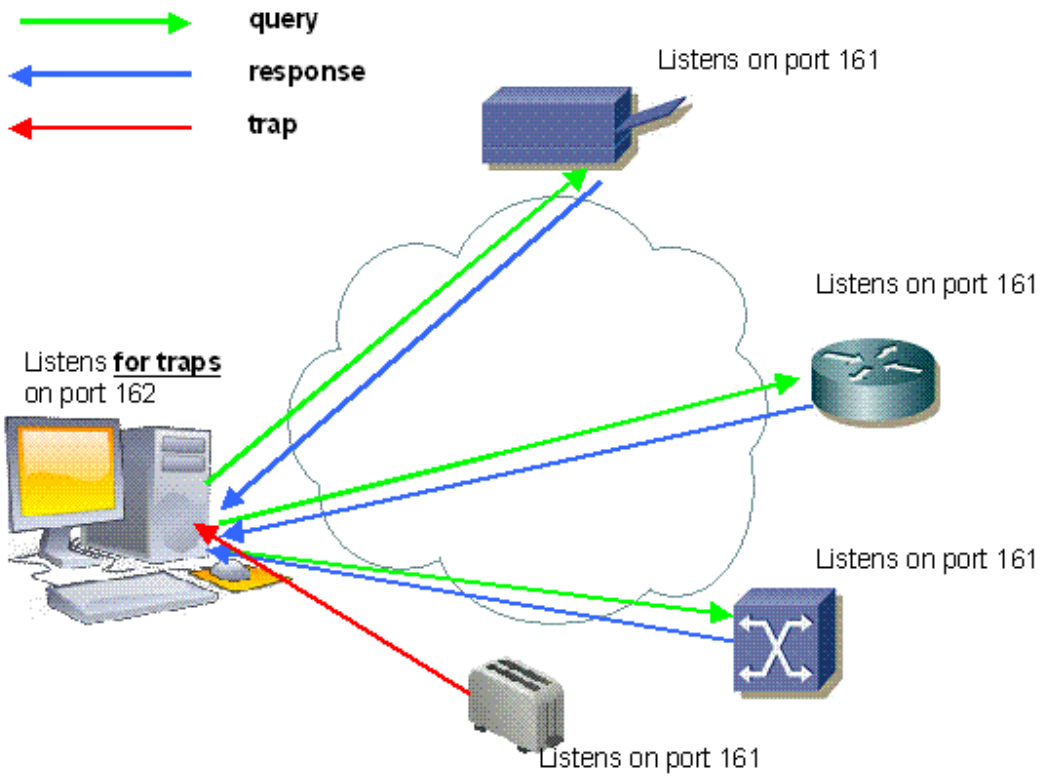
Agent, mis asub igas hallatavas seadmes, on võimeline vastu võtma, saatma ja töötlema SNMP teateid. Agent on vahelüli seadme informatsiooni ja NMS päringute vahel. NMS-lt saadetud teate peale otsib agent vajaliku info seadmest ja saadab selle haldusjaamale tagasi. Lisaks on agent võimeline kirjutama NMS-i poolt saadetud konfiguratsiooni muudatusi seadmesse. Agendi poolt sooritatava informatsiooni lugemise ja kirjutamise õigusi ühe või mitme NMS-i jaoks määratakse seadmes. Need õigused määratakse ära agendi pääsu reguleerimise seadistustes ja määravadki siis missugused NMS-id üldse saavad pärida antud seadme informatsiooni ja kui saavadki pärida, siis kas neil on õigus ka SNMP kaudu seadme konfiguratsiooni muuta. [6]

1.1.2 Üksuste vaheline suhtlus

SNMP kasutab klient- server arhitektuuri. Sellise arhitektuuri puhul algatab klient suhtluse saates päringu teenuse või ühenduse loomiseks. Server ootab passiivselt selliseid

päringuid. SNMP, olles interneti protokollikomplekti liige (kutsutakse TCP/IP protokollistikuks, isegi kui transpordikiht ei ole TCP vaid UDP nagu see on SNMP puhul), kasutab üldtuntud pordinumbreid serveri poolel, et teha kindlaks rakenduskihi protokolli. [6]

Jooniselt 1 on näha, et kuna NMS algatab SNMP päringu, samal ajal kui agent erinevates hallatavates seadmetes (pildi peal ülevalt allapoole vastavalt näha printer, marsruuter, kommutaator ja röster, millest eelnevalt juttu oli) passiivselt ootab päringuid, siis saab öelda, et võrguhaldusjaam ise on klient ja agent on server. Port 161 on üldtuntud pordinumber, mis on määratud SNMP rakendusele- seega seadmetes olevad agendid kuulavad pordil 161 NMS-i poolt saadetud teateid (mõned tootjad lubavad SNMP standardporti 161 agendi seadistustes muuta. Kui seda porti seadmes muuta, siis tuleb ka NMS-i muudatustest teavitada ehk määrata päringute teostamisel mingi muu port, kuhu poole SNMP teated saata [2, lk 20]). SNMP kasutab tegelikult kahte klient- server rakendust: SNMP päring/ vastus protokolli ja SNMP *trap* protokolli. *Trap*'id, nagu juba punktis „1.1.1 Üksused" mainitud, saadetakse agentide poolt ja selle eesmärk on saata hoiatusteade mingi sündmuse kohta. Kuna agent ise on *trap*'i algataja, siis antud juhul ei saa öelda, et agent käituks serverina kui ta saadab ise teate. Sel juhul käitub agent kliendina ja NMS, mis on seadistatud *trap*'e vastu võtma hallatavatelt seadmetelt, käitub serverina kuulates teisel üldtuntud pordil, milleks on port 162, *trap*'i teateid. IANA üldtuntud portide nimekirjas on näha, et port 161 on määratud SNMP jaoks ja port 162 on määratud SNMPTRAP jaoks. Seega käitubki NMS tavaliselt kliendina, kuid mõnikord ka serverina ja agent käitub tüüpiliselt serverina, kuid samas aeg- ajalt ka kliendina. [6]



Joonis 1. SNMP üksuste vaheline suhtlus [9]

Samas, erinevalt tüüpilistest kliendirakendustest, kui räägime *trap*'i rakendusest, ei oota agent mingisugust vastust- vähemalt mitte teatekujulist vastust. *Trap*'ile võib järgneda inimese poolne reageering, kuid SNMP kavandati nõudmaks minimaalseid agendipoolseid ressursse (seade peab ikkagi tegema ja andma ressursse eelkõige selle tarvis, mille jaoks ta on mõeldud ehk siis näiteks marsruuteri puhul liikluse õigesse kohta marsruutimiseks) ja seega ei oota ta vastust ning kinnitust, et *trap* on võrguhaldusjaama poolt vastu võetud. Agent saadab *trap*'i välja lootes, et NMS kuulab 162 pordil ja võtab teate vastu (siiski on defineeritud SNMP-le hiljem ka tagasiside tüüpi *trap*, mida nimetatakse InformRequest. Selle puhul saadetakse *trap*'i saatjale tagasi sõnum, et teade on vastu võetud [8]). [6]

Võrguhaldusjaama ja agendi vahelise transpordiprotokollina kasutab SNMP protokolli UDP. UDP kasuks TCP ees otsustas asjaolu, et UDP on ühenduseta, mis tähendab seda, et lõpp- punktide vahelist ühendust ei looda NMS-i ja agendi vahel kui andmepakette

saadetakse edasi ja tagasi. See aspekt teeb UDP ebausaldusväärseks kuna kadunud andmepakettide kohta protokolli tasandil ei ole mingeid tõendeid. Seega jääb SNMP rakenduse ülesandeks kindlaks määrata, kas andmepaketid on kadunud ja soovi korral need uuesti saata. Tavaliselt tehakse seda lihtsa sõnumi aegumisega (*timeout*). Sõnumi aegumist on võimalik NMS-s seadistada. NMS saadab UDP päringu agendile ja ootab vastust ning kui sõnumi vastus viibib liialt kaua ja NMS pole agendilt vastust saanud, eeldab võrguhaldusjaam, et andmepakett läks ülekandel kaduma. Päring saadetakse agendile uuesti ning see, mitu korda päring saadetakse, on samuti seadistatav. [2, lk 19]

UDP põhimõte SNMP kasutamise juures ongi seega minimeerida agendi ressursse võrguhalduse joaks kuna liiasus (*overhead*) on väike. SNMP-d kasutatakse ka mõningates spetsiifilistes olukordades TCP-ga, kuid üldjuhul SNMP töötamine üle TCP on halb idee. Oluline siinkohal on mõista, et usaldusväärne TCP ei ole alati parim lahendus ja SNMP on tehtud töötamaks võrkudes, kus esineb probleem. Kui infrastruktuur töötaks alati laitmatult, siis poleks seda vaja jälgidagi. Kui mõni seade võrgus ei tööta, siis on protokoll, mis üritab saada seadmelt infot kätte, kuid lõpuks annab alla (ehk siis UDP), igal juhul parem lahendus kui protokoll, mis ummistab võrku korduvate kontrollteadetega püüdluses saavutada usaldusväärsus (ehk siis TCP). [2, lk 19]

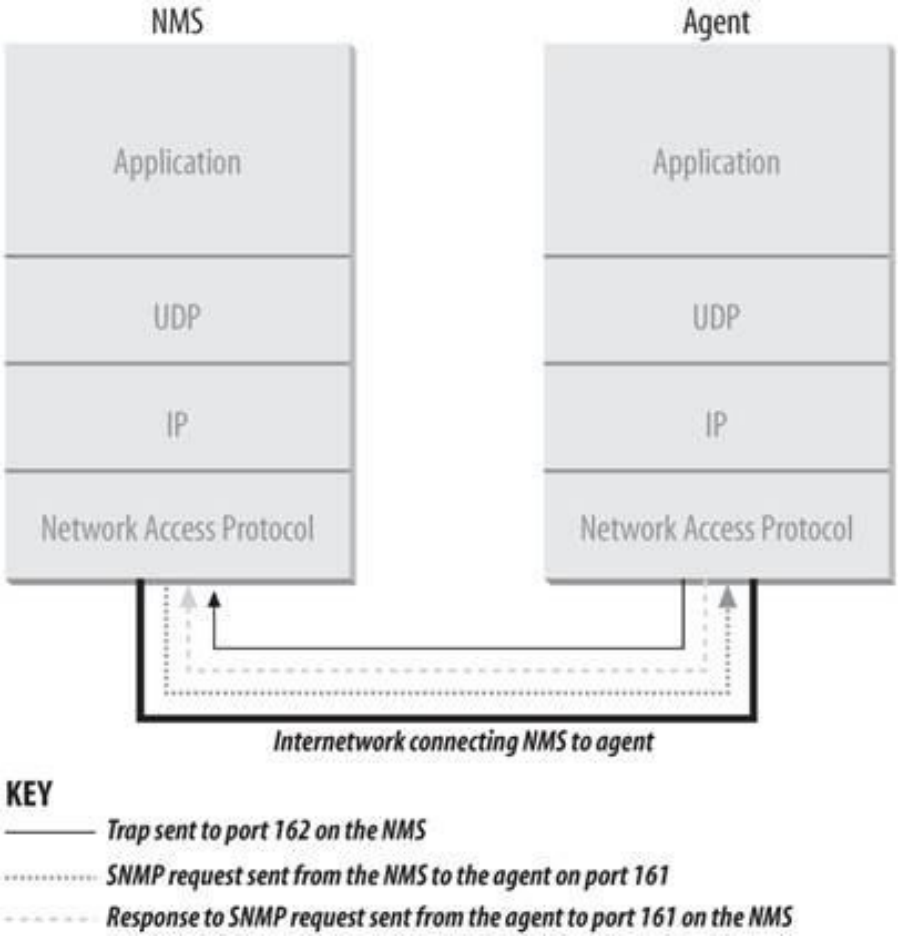
SNMP teadete saatmine TCP/IP protokolli mudelis on põhimõttelise skeemina kujutatud joonisel 2. Kui NMS või agent soovib teostada SNMP toimingu (päring tegemine või *trap*'i saatmine), siis toimuvad järgmised sündmused protokollipinus:

rakenduskihil (*Application*) – esmalt otsustab reaalne SNMP rakendus (NMS või agent) mida ta soovib teha (saata päring, saata päringule vastus või edastada *trap*). Rakenduskiht tagab teenused lõppkasutajale, näiteks administraator pärib Ethernet kommutaatori pordi staatuse informatsiooni.

transpordikihil (UDP) – lubab kahel arvutil omavahel suhelda. UDP päis sisaldab, lisaks teistele asjadele, sihtkoha seadme porti, kuhu saadetakse päring või *trap*. Sihtkoha port on sel juhul vastavalt kas 161 või 162.

võrgukihil (IP) – SNMP pakett saadetakse määratud sihtkohta, mis on näidatud IP aadressiga.

andmelülikihil (*Network Access Control*) – viimane sündmus, mis SNMP paketiga tehakse, et ta jõuaks sihtkohta, on anda see üle füüsilisele võrgule, kus see suunatakse lõplikkusse sihtkohta. Andmelülikiht koosneb riistavarast ja seadmedraiverist, mis paneb andmed füüsilisele traadile. Andmelülikiht on vastutav ka pakettide vastuvõtmise eest füüsilisest võrgust ning nende saatmise eest tagasi üles protokollipinus, et pakette saaks töödelda rakenduskihi poolt (antud juhul SNMP).



Joonis 2. SNMP teadete vahetus TCP/IP protokollistikus [2, lk 20]

Samuti tasub märkida, nagu eelpool mainitud, et SNMP töötab ka OSI arhitektuuriga võrkudes, kus UDP transport pole võimalik [3]. Samuti suudab SNMP töötada *AppleTalk* [4] ja IPX [5] võrkudes.

1.1.3 Haldusinfostruktuur

Eelnevalt on kirjutatud, et SNMP tuumikuks on tegevuste kogumik (ja loomulikult informatsioon, mida nende tegevuste käigus kogutakse), mis annab võrgu haldajale võimaluse pärida ja sealjuures ka muuta SNMP-d toetavate seadmete objektide olekut. Väga oluline osa SNMP toimimise juures ongi haldusinfo ehk siis võrguhalduse informatsioon, mis liigub NMS-i ja SNMP-d toetatavate seadmete vahel. Haldusinfo koosneb hallatavatest objektidest (*managed object*), mille kohta edaspidi öeldakse ka lihtsalt objekt. Terminit objekt võibki defineerida kui haldusinfo ühik või kui haldusinfo üks tükk. Samas ei tohi segamini ajada hallatavat objekti ja hallatavat seadet või tarkvara, sest need on erinevad asjad. Objekt siin kontekstis on abstraktne mõiste, samas kui seade või tarkvara on reaalselt eksisteeriv riistvara või programm. [6]

Objektide täpne kirjeldamine, nimetamine ja objektidega seotud andmetüübid määratakse haldusinfostruktuuri (*Structure of Management Information* ehk edaspidi SMI) poolt. Õigemini määrati need ära SMI versioon 1 (SMIv1) poolt, SMI versioon 2 (SMIv2) on haldusinfostruktuuri edasiarendus ja seda SNMP versioon 2 jaoks (kasutusel olev SMI versioon ei ole versiooni numbri poolest seotud kasutatava SNMP versiooniga). Objekti määratlemiseks saab selle põhimõtteliselt jagada kolmeks osaks:

nimi – objekti identifikaator (*object identifier* ehk edaspidi OID) määrab ära unikaalselt hallatava objekti ehk igale objektile vastab kindel ning kordumatu OID. Objekti identifikaator on kahel kujul: numbrilisel ja inimloetaval ehk tekstilisel esitlusel.

tüüp ja süntaks – objekti andmetüüp on defineeritud kasutades *Abstract Syntax Notation One* (ASN.1) alajaotist. ASN.1 määrab ära, kuidas SNMP maailmas andmeid esitletakse ja edastatakse NMS-i ja agentide vahel. ASN.1 suur pluss on tema sümboolika sõltumatus (nagu näiteks baitide järjekord), mis tähendab, et personaalarvuti, mille peal on Windows 7 operatsioonisüsteem saab vabalt suhelda Sun SPARC masinaga ilma, et tekiks probleeme.

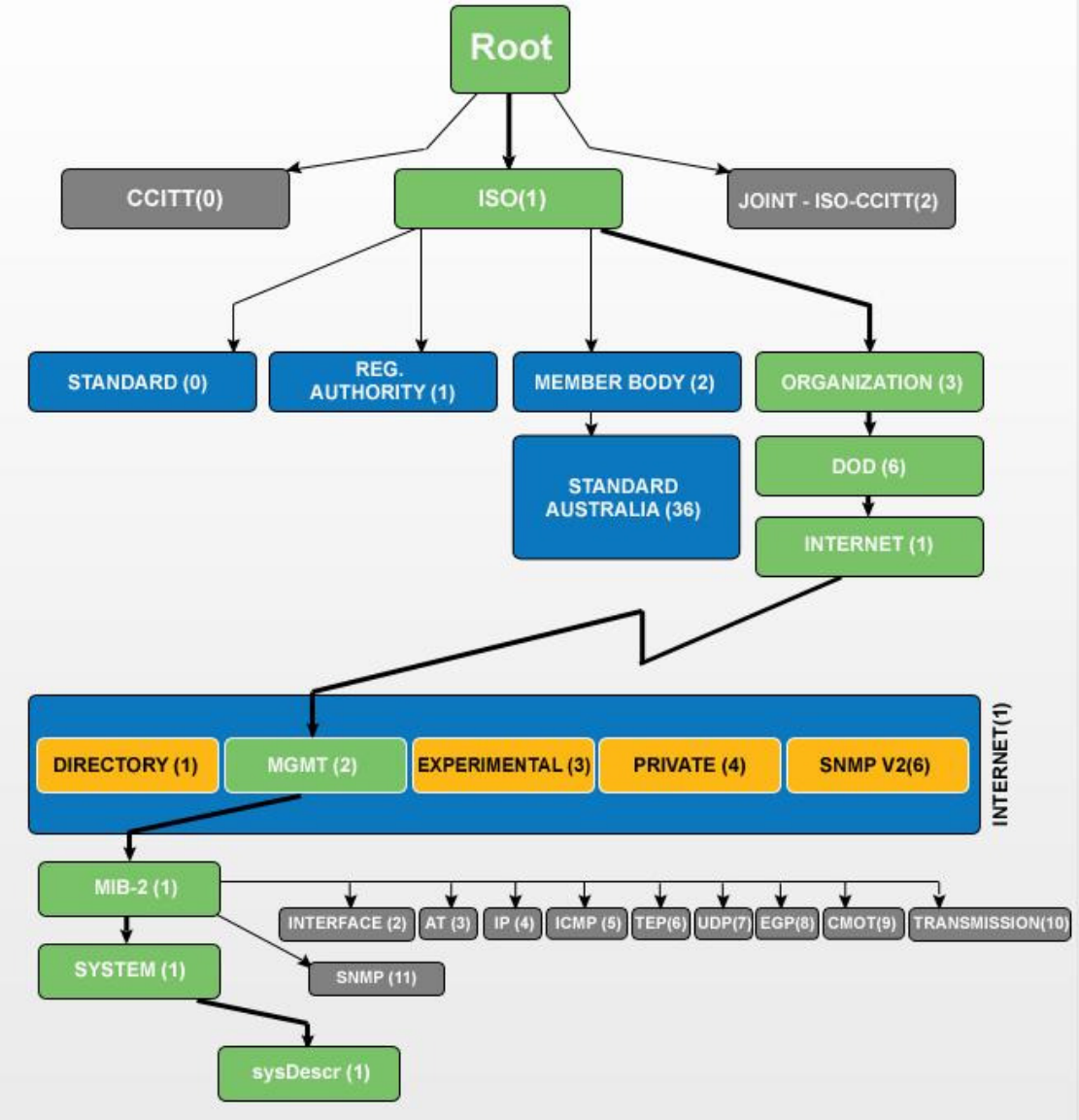
kodeerimine – hallatava objekti üks eksemplar kodeeritakse oktettide sõneks (*string of octets*) kasutades põhilisi kodeerimise reegleid (*Basic Encoding Rules* ehk BER). Kodeerimisreeglid määravad, kuidas objektid kodeeritakse ja dekodeeritakse nii, et neid

saaks saata üle edastuskeskkonna nagu näiteks Ethernet. BER kodeerimise osa pikemalt antud magistritöös ei käsitleta. [2, lk 23]

SNMP info saamiseks päritaksegi seadmest objekte, kuid selleks, et teada missuguseid objekte otsida ja kuidas neid üles leida, peavad olema kindlad reeglid ja struktuur. Sellise struktuuri saavutamiseks ongi objektid organiseeritud puulaadsesse hierarhiasse, mis on näha joonisel 3. Objekt koosneb seega täisarvude (*integer*) jadast baseerudes joonisel 3 näha olevast puu sõlmpunktidest, mis on eraldatud punktidega. On olemas ka inimloetav kuju, kuid sel juhul koosneb OID puu sõlmpunktide nimede jadast, mis on samuti kõik eraldatud punktidega. Seega saab kasutada numbreid kui ka numbritele vastavaid nimesid, et moodustada kindel objekt. [2, lk 24]

Objektipuus kutsutakse kõige ülemist sõlme (*node*) juur (*root)*, sõlme, millel on oma alamsõlmed, kutsutakse alampuu sõlmeks (*subtree*) ning sõlme, millel ei ole alamsõlmi, kutsutakse leht sõlmeks (*leaf node*). Näiteks on joonisel 3 iso(1) alampuu sõlm, kuid ccitt(0) ja joint(2) leht sõlmed. [2, lk 25]

Käesolevas magistritöös on fookus *iso*(1).*org*(3).*dod*(6).internet(1) alampuul, mis kirjeldab OID-d numbrilisel kujul 1.3.6.1 või teistmoodi tekstilisel kujul *iso.org.dod.internet*. Fookus on sellel alampuul kuna just *iso.org.dod.internet* all on kirjeldatud kõik SNMP objektidega seonduv informatsioon. Sealt edasi on lähema vaatluse all OID 1.3.6.1 alampuud mgmt(2).MIB-2(1), mis kirjeldab standardsed üldkasutatavad objektid, ja *private* (4), mis määrab juba tootjaspetsiifilised objektid. Nagu juba mainitud, siis on igal hallataval objektil numbriline OID kuju ja sellega seotud tekstiline nimi. Numbriline kuju määrab ära selle kuidas objekt on kirjeldatud sisemiselt agendis. Tekstiline kuju, mida võib võrrelda kui arvuti täieliku domeeninimega, on mõeldud inimeste elu lihtsamaks tegemiseks, et nad ei peaks meelde jätma pikki ja igavaid täisarvude jadasid. [2, lk 25]



Joonis 3. Haldusinfostruktuuri objektide puu [45]

Oluline on ka teada, et agendiga süsteemis hoitakse haldusinfot kahel kindlal kujul, kuid selle info transportimisel näeb ta välja hoopis teistsugune. Seadmes hoitakse infot siis kahel viisil: skalaarsel- ja tabelkujul. Skalaarne tähendab, et objekt on ühemõõtmeline ja objekti väärtust väljendatakse arvuga või sõnega. Tabel tähendab, et objekt on kahemõõtmeline massiiv, kus objekti väärtusi (ehk siis tabeli puhul on objektiks tabel ise) võib olla väga palju vastavalt tabeli ridade ja veergude arvule. Transportimisel aga saab infot vahetada ainult skalaarsel kujul ehk siis tervet tabelit korraga ei ole võimalik SNMP

puhul edastada ja igat tabeli elementi tulebki saata eraldi infona (samas on tulnud võimalus saata SNMP versiooni 2 ja 3 puhul mitu tabeli elementi ühe paketi sees. Selle kohta on täpsemalt kirjas lisas 3). [11]

Skalaarsete objektide reaalse info kättesaamiseks tuleb pärida vastavat OID-d ja lõppu panna ka number 0. Näiteks pärides joonisel 3 näha olevat sysDescr(1) OID-d, siis OID enda numbriline ülesmärkimine oleks .1.3.6.1.2.1.1.1. Lihtsalt selle numbriga pärides oleks tulemuseks veateade. Reaalse väärtuse saamiseks tuleb päring esitada kujul .1.3.6.1.2.1.1.1.0 ning SNMP abil saadakse teada päritava süsteemi nimi. Tabelobjekti puhul tabelist reaalse info kättesaamiseks kasutatakse mõistet indeks (*index*), mis põhimõtteliselt tähendab esimest veergu või siis esimese veeru väärtusi. Näiteks võivad tabeli esimese veeru väärtused tähistada seadme võrguliideseid. Kui seadmel on 24 võrguliidest, siis seega on tabelil 24 rida ja iga rea esimene väärtus (ehk põhimõtteliselt võrguliidese number) on ka indeksiks. Kindlasti on tabelil ka rohkem veerge kui üks, mis võivad tähistada liidese kirjeldust, töösolekut, vastuvõetud pakette, saadetud pakette jne. Kui nüüd on soov tabeli objektist mingi kindla liidese kohta teatud informatsiooni saada, siis tehakse päring kujul OID.tabeli_veerg.tabeli_indeks (siin enam 0 lõppu ei lisata). Näiteks on vaja teada saada neljanda liidese kirjeldust (arvestusega, et kirjeldus liideste kohta asub tabeli teises veerus). Sel juhul oleks SNMP päring kujul: tabelobjekti OID.2.4. [11]

Objekti definitsioon sisaldab nelja osa:
SYNTAX – määrab ära, millise andmetüübiga tegu on (sõne, täisarv, loendur jne).
MAX-ACCESS – defineerib maksimaalse ligipääsu objektile ehk kas objekti ei saa üldse pärida, saab pärida objekti väärtusi või saab nii pärida kui ka seada objekti väärtusi.
STATUS – näitab objekti staatust (kas on hetkel kasutusel, kaob varsti kasutuselt või on juba vana objekt, mida enam ei kasutata).
DESCRIPTION – objekti kirjeldus. Selle lõpus on ka objekti unikaalne identifikaator [11]

Reaalne objekti defineerimine joonisel 3 näha oleva sysDescr(1) objekti kohta SNMPv2-MIB-is näeb välja järgnevalt:

```
sysDescr OBJECT-TYPE
        SYNTAX      DisplayString (SIZE (0..255))
        MAX-ACCESS  read-only
        STATUS      current
        DESCRIPTION
            "A textual description of the entity.  This value should
            include the full name and version identification of
            the system's hardware type, software operating-system,
            and networking software."
        ::= { system 1 }
```

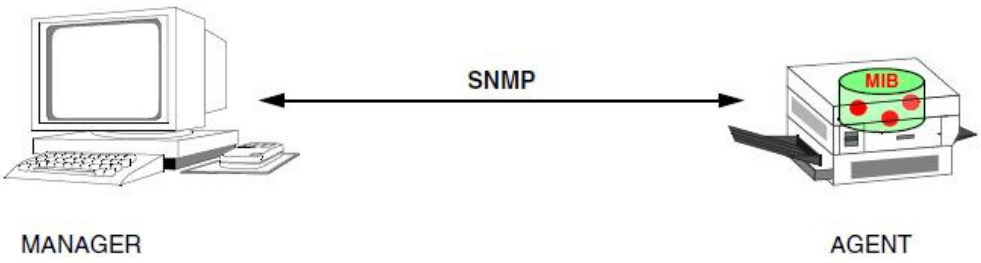
1.1.4 Haldusinfobaas

Kõik haldusinfo objektid on organiseeritud kindlate tingimuste alusel ja pandud kirja baasidesse. Neid baase kutsutakse haldusinfobaasi mooduliteks ehk *Management Information Base module* (edaspidi kasutatakse ka lühendit MIB). On olemas standardsed MIB moodulid, mida toetavad kõik seadmed, mis toetavad SNMP-d. Samuti on olemas standardsed MIB failid, mida toetavad seadmed, mille jaoks kindel standardne haldusinfobaas on oluline. Lisaks on olemas tootjaspetsiifilised privaatsed MIB moodulid, mis sisaldavad objekte, mis on kasutusel ainult selle tootja seadmetes. Näiteks peab olema kõikides SNMP poolt hallatavates seadmetes mingi kindel informatsioon, milleks võiks olla IP aadress. Selline informatsioon ongi määratud standardses MIB-is, mida peavad toetama kõik seadmed, mis soovivad, et neid arvestataks SNMP-ga ühilduvaks. Kui seadmel on Etherneti liides, siis on mingi informatsioon, mida antud seade peaks suutma väljastada, näiteks pakettide kokkupõrgete (*collision*) number. Selline informatsioon on määratud standardses Etherneti MIB-is ja seda peaksid toetama kõik seadmed, millel on vastav liides. Lisaks võib kindel tootja väljastada infot seadmes töötava ventilaatori kohta. See on juba väga tootja- ja tootespetsiifiline info ning sel juhul

peab kindel seade ka toetama MIB-i, mille abil saab kätte ventilaatori pöörete sageduse. Sellised MIB-id on privaatsed ja üldiselt ei ole standardiseeritud. [6]

MIB-ist võib seega põhimõtteliselt mõelda kui hallatavate objektide andmebaasist, mis esindab süsteemi ressursse ja mida agent jälgib (reaalselt MIB siiski ei ole objektide andmebaas, sest andmebaas sisaldab ka tegelikke andmeid. MIB on pigem andmebaasi struktuur. Andmed ise on riistvara või tarkvara poolt paigutatud kindlates kohtades. Agent lihtsalt teab, kuidas andmeid nendest kohtadest välja otsida või saata käsk mingit väärtust muuta [12]). Kogu seadme või tarkvara info, millele on võimalik NMS-i abil ligi pääseda, on defineeritud MIB-ides. SMI määrab, kuidas hallatavad objektid peavad välja nägema ja MIB näitab, missugused objektid üldse olemas on (kasutades SMI süntaksit) ning SNMP sealjuures määrab, kuidas haldusinfot ehk objekte vahetatakse üle võrgu. MIB-i analoogina saab tuua sõnaraamatut, kus siis defineeritakse sõna (MIB-is defineeritakse hallatav objekt) ja seejärel seletatakse sõna tähendust (ehk siis MIB-is on objekti juures kirjas, mida saab antud OID-ga teha). [2, lk 4]

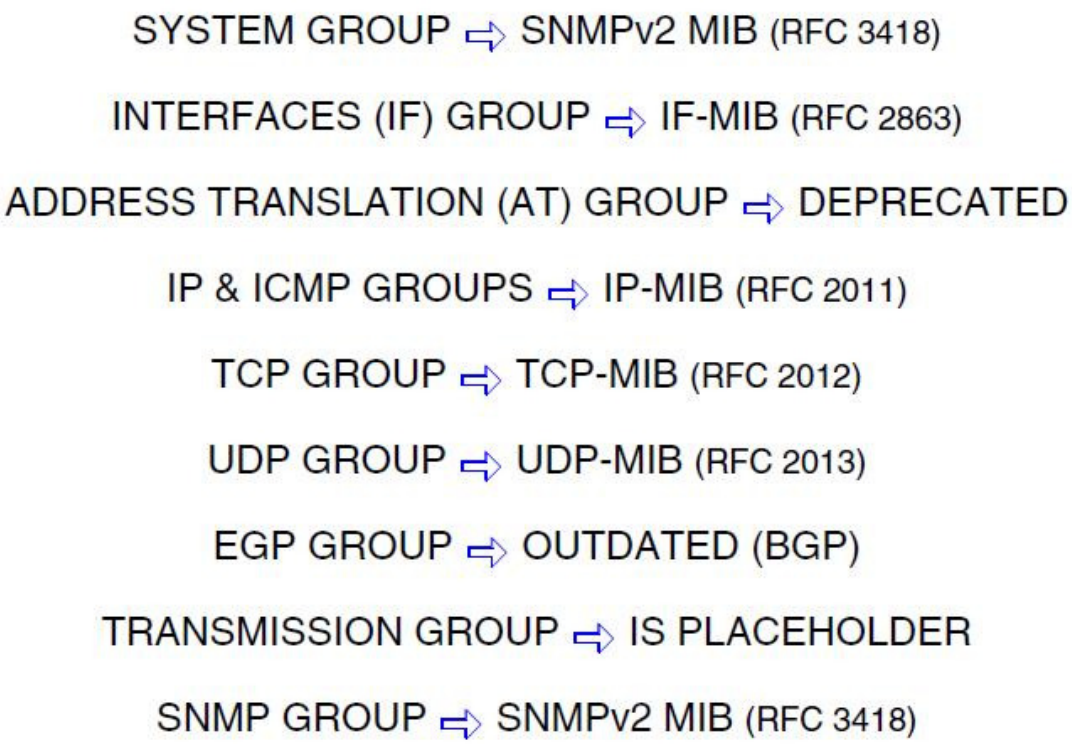
MIB-ist paremaks arusaamiseks on joonis 4, kus siis seadmes asubki haldusinfobaas. Samas peaks olema vastav MIB fail ka NMS masinas, sest selle abil on väga kergesti võimalik haldajal teada saada, mis objektid asuvad hallatavas seadmes.



Joonis 4. Haldusinfobaas seadmes [13]

Vahest kõige olulisem haldusinfobaas on MIB-2, sest iga seade, mis toetab SNMP-d peab toetama ka seda MIB-i. MIB-2(1) alampuu on näha joonisel 3 (asub leheküljel 23). Nagu ka jooniselt näha, siis MIB-2 koosneb mitmest erinevast rühmast ja need rühmad on lahti

võetud ning defineeritud erinevates MIB moodulites, mis baseeruvad SMIv2 peal. MIB-2 määrab ära objektid, mille abil saab jälgida ja muuta seadme süsteemi kirjeldusi, liideste informatsiooni ja võrguliikluse andmeid. Kokku on umbes 170 OID-d, millest enamuse väärtused on ainult loetavad, üksikutel objektidel on võimalik ka väärtusi seadistada. Seda, millise rühma hallatavad objektid asuvad millises haldusinfobaasi moodulis, on näha joonisel 5. Lisaks on eelmainitud joonisel näha ka RFC number, mis vastab kindla MIB-2 rühma MIB moodulile. [15]

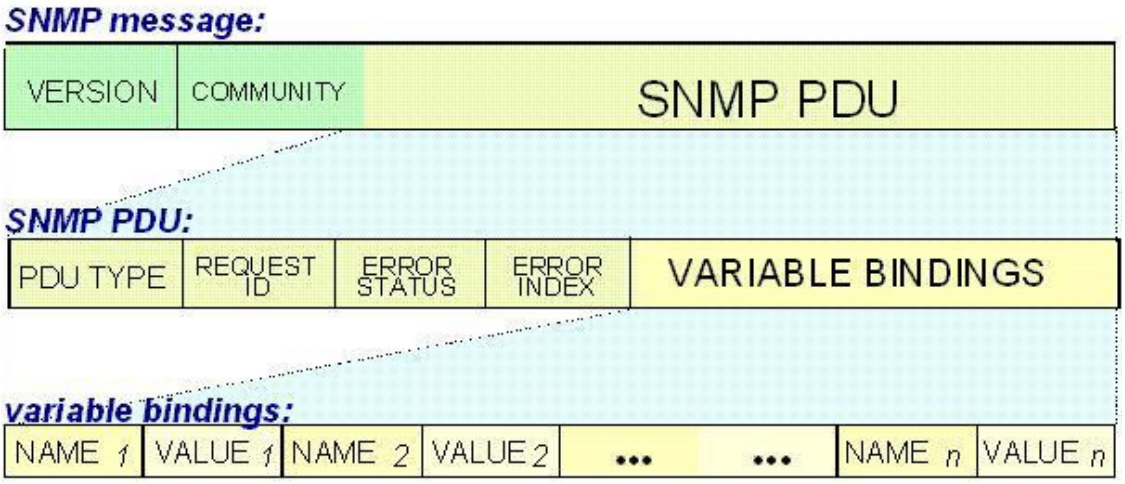


Joonis 5. MIB-2 rühmad ja neile vastavad MIB moodulid [14]

1.2 Protsessid

Eelnevalt on tutvustatud SNMP komponente, kuidas nad on organiseeritud ja mis on erinevate osade ülesanded, kuid missugused on protokolli protsessid (*operations*) ja kuidas üldse informatsiooni kogutakse, tuleb vaatluse alla käesolevas paragrahvis.

Protokolli protsessid SNMP-s kapseldatakse protokolliandmeüksustesse ehk PDU-desse, mis omakorda kapseldatakse sõnumi (*message*) päisesse. Antud juhul tutvustab autor SNMP versiooni 2 PDU formaati kuna see asendab versiooni 1 formaati ning võrreldes versiooni 1 formaadiga on versiooni 2 PDU formaati täiustatud. SNMP versiooni 3 muudatusi tutvustatakse paragrahvis „1.3 Turvalisus“. Versiooni 2 üldine sõnumi formaat ongi kujutatud joonisel 6, kus ülemine osa kujutab tervet SNMP sõnumit, joonise keskmine osa kujutab PDU formaadi lahtikirjutist ja alumine osa näitab objekti muutujate seotust ehk siis objekti nime ja tema väärtuse paari (*variable bindings*). [16]



Joonis 6. SNMP versiooni 2 sõnumi formaat [16]

Sõnumi päis sisaldab kahte välja lisaks PDU-le: versioon ja kogukond (*community*), millest viimane on põhimõtteliselt turvalisuse mudel turvalisust tagamata (kogukonda on täpsemalt kirjeldatud paragrahvis „1.3 Turvalisus“). PDU sisaldab nelja välja lisaks objekti ja väärtuse paaridele: PDU tüüp (PDU *type*), päringu ID (*Request* ID), vea staatus (*Error status*) ja vea indeks (*Error index*). PDU tüüp näitab millist protsessi tüüpi sõnum täidab, näiteks 0 on *get* ja 2 on *response*. Päringu ID on number, mida kasutatakse päringute ja vastuste võrdlemiseks. Päringu ID luuakse seadme poolt, mis saadab päringu ja kopeeritakse samale väljale *response*-PDU-s vastust andva agendi poolt. Vea staatus ja vea indeks väljad on väärtusega 0, kui mingit viga ei esine. Juhul kui agendil oli viga paketi töötlemisel, siis vea staatuse väljal on pandud mingi väärtus, näiteks 17 ehk *notWritable*, mis lisatakse, kui haldaja üritab kirjutada seadmesse teatud objektile mingit

väärtust, mida tegelikult saab ainult lugeda. Kui vea staatus on nullist erinev ja samuti vea indeks on nullist erinev, siis indeks viitab nimekirjas olevale objekti ja tema väärtuse paarile, millega on probleem, kõik teised objekti ja väärtuse paarid on õiged. Juhul kui vea staatus on nullist erinev ehk mingi asjaga on probleem ja vea indeks on 0, siis tähendab see, et kõik objekti ja väärtuse paarid on vigased (vigast objekti ja väärtuse paari ei suudeta sel juhul tõenäoliselt täpselt ära näidata). Vea staatus ja vea indeks välja kasutus on SNMP versioonis 1 natuke teistsugune. Nimelt on SNMP versiooni 2 puhul edasi arendatud veatöötlust, kaasa arvatud detailsemad veakoodid. Näiteks SNMP versiooni 1 puhul viitab veakood *badValue*'le, kuid SNMP versiooni 2 agent suudab olla üksikasjalikum ja valida *badValue* asemel *wrongValue, wrongEncoding, wrongLength* jne. [16]

PDU-s sisaldav haldusinfo asub objekti ja väärtuse paaride väljal. Päringut tehes on objekti osa täidetud OID-ga, kuid väärtus osa täidetud *null*-iga (see viitab tühjale väärtusele), kuid vastuses on loomulikult igale objektile pandud juurte ka reaalne väärtus. Mitme objekti ja väärtuse paari lubamine PDU-s tähendab, et ühe sõnumiga on võimalik saada mitme objekti väärtus, mis on väga positiivne. [16]

Eelnevalt on mainitud, et PDU on sõnumi formaat, mida võrguhaldusjaamad ja agendid vahetavad, et saata ja vastu võtta haldusinformatsiooni. Igal SNMP protsessil on oma kindel PDU formaat. SNMP protsessid ise on järgmised: *get, getnext, getbulk, set, response, trap, notification, inform, report*. Lisas 3 on toodud SNMP protsesside toimimise kirjeldused ja nende PDU formaadid. [2, lk 37]

1.3 Turvalisus

Antud paragrahv on kirjutatud allika [17] põhjal.

SNMP turvalisuse kirjeldamiseks tuleks kõigepealt lahti seletada mõningad ohud, mis teatud juhtudel võimaldavad halbadel eesmärkidel ära kasutada võrguhaldusjaamade ja

agentide suhtluses vahetatavat võrguhaldusinformatsiooni. Lihtsamalt öeldes tekib ohtude realiseerumisel informatsiooni turvalisuse kadu. Ohtude kirjeldamisel on nende juures ka öeldud, mis mõju nad võrguhaldusele üldises mõttes avaldavad. Siinjuures tuleks ka ära märkida, et turvalisus koosneb kolmest komponendist: käideldavus ehk informatsiooni õigeaegne ning mugav kättesaadavus, terviklus ehk informatsiooni pärinemine autentsest allikast ning veendumine, et informatsioon pole hiljem muutunud või neid pole hiljem volitamatult muudetud ja konfidentsiaalsus ehk informatsiooni kättesaadavus ainult selleks volitatud isikutele.

Teesklus (*masquerading*) – oht, mille käigus ründajal õnnestub esineda kellegi teise rollis ja viia läbi mingeid ülesandeid ohvri nimel. Võrguhalduse turvalisuse seisukohast on see ehk kõige tõsisem oht. Üks viis kasutada teesklust on tüssamine (*spoofing*), mille käigus võltsitakse saatja aadressi. Kui pahatahtlikul ründajal õnnestub esineda volitatud haldajana, siis on tal kõik õigused, mis haldajale määratud. Kui volitatud haldajale on antud kõik õigused süsteemi kasutamiseks, siis nüüd saab need õigused endale ka ründaja.

Informatsiooni muutmine (*modification of information*) – oht, mille puhul õnnestub kolmandal osapoolel sõnumi edastus kinni püüda ja kuritahtlikult selle sisu muuta. Seejärel saadetakse muudetud sõnum edasi sellele, kes selle pidi ka algselt saama. Saaja arvabki, et sõnum saadeti usaldusväärse saatja poolt, kuigi sõnumi sisu on hoopis teistsugune, mis ta oli algsel saatmisel. Võrguhalduse seisukohast võib küll volitatud administraator luua õige PDU, kuid kui pahatahtlik osapool selle sõnumi vahelt haarab ja PDU ära muudab, jättes autentimise (näiteks kogukonna nime) informatsiooni samaks, siis pole vastuvõtval poolel põhjust kahelda sõnumi tõepärasuses. Muidugi on võimalik sellise ohu realiseerumisel ainult siis kui PDU ei ole allkirjastatud või krüpteeritud.

Sõnumivoo muutmine (*message stream modification*) – oht, kus sõnumite voog on kuidagi muudetud. See tähendab, et sõnumite järjekorda on muudetud või sõnumid on salvestatud ja uuesti saadetud. Kuna enamus haldusprotokolle disainiti töötama ühenduseta transporditeenustel, siis on sõnumivoo muutmine tõsine oht võrguhalduses. Näiteks võib ründaja salvestada õige sõnumi, mis käsib marsruuteril ennast välja lülitada.

Millalgi hiljem võib ründaja seda sama salvestatud sõnumit kasutada seda uuesti marsruuterile saates ja sellega seadme välja lülitada hetkel, mil seda kindlasti teha ei tohiks.

Avalikustamine (*disclosure*) – oht, mille puhul salajased andmed lekitatakse inimestele, kes ei peaks seda nägema. Võrguturvalisusest üldiselt rääkides on üks selliseid viise krüpteerimata võrguliikluse nuuskimine (*sniffing*). Selline liiklus võib aga sisaldada väga olulist informatsiooni võrgu ja hallatavate seadmete kohta. Kui ründaja peaks olulise informatsiooni enda kätte saama, siis võib ta seda kasutada enda kasuks või selle abil sooritada ründeid. Avalikustamise vastu võitlemisel on laheduseks võrgus liikuvate sõnumite krüpteerimine.

Teenusetõkestamine (*Denial of Service* ehk DoS) – tähendab, et mingi võrguteenus blokeeritakse. Ründaja võib näiteks üritada avada järjepidevalt TCP ühendusi seadmel takistades sellega teiste ühenduste loomist. Võrguhalduse seisukohalt tähendaks see, et ründajal õnnestub blokeerida haldussõnumite saatmine haldusjaama ja agentide vahel. Samuti võib olla teenusetõkestamine mingi teise ründe tagajärg. Näiteks kasutab ründaja teesklust ja näitab ennast kui administraator. Administraatori õigustega saab ta marsruuteri välja lülitada ja see tegelikult ongi juba teenustõkestamise rünne kuna võrguteenused ei tööta. Teenusetõkestamise ohtu on väga raske ennetada ja erinevalt eelnevatest kirjeldatud ohtudest ei ole SNMP-l selle ründe puhul mingisugust kaitset.

1.3.1 SNMP versioon 1

Algelisel SNMP versioonil (SNMP versiooni 1 kohta kasutatakse ka edaspidi lühendit SNMPv1) on väga primitiivsed turvalisuse funktsioonid. Ainuke viis agendil NMS-i autentida ehk kontrollida kas päring tuleb ikka sealt, kust ta peaks tulema, on kogukonnanime järgi. Kogukonnanime kasutatakse selleks, et määrata kas ja missugusel võrguhaldusjaamal on õigus *get* ja *set* protsessidega agendilt infot pärida ja seada. Samuti

kasutatakse seda, et anda erinevatele NMS-idele ja samas ka MIB-i moodulitele ligipääsuõigused.

SNMP kogukond määratakse lokaalselt seadmes ja seda sama nime saab kasutada ka mitmetes erinevates seadmetes. Vaikesätetena on seadmetes kogukonnanimi üldiselt *public*. Kui NMS soovib läbi viia mõnda halduskäsku (*get* või *set*), siis peab ta alati esitama kogukonnanime. Selleks, et hallata tervet hulka seadmeid peab võrgu haldaja säilitama nimekirja olulisi kogukonnanimesid.

Sellist kogukonnapõhist autentimisviisi ei saa pidada turvaliseks. Igaüks, kellel on teada kogukonnanimi, võib esineda kui süsteemi haldaja. Kuna kõikidel seadmetel võib olla ka sama kogukond määratud, siis võib olla kuritahtlikul inimesel väga suur võim hallatavate seadmete üle. Teine suur probleem SNMPv1-ga on asjaolu, et puudub täielik informatsiooni privaatsus kuna pole võimalik haldusinfosõnumeid krüpteerida. Kui informatsioon liigub läbi turvamata avaliku võrgu, siis ei saa keegi öelda kas liiklust kuulatakse pealt või mitte. Seega on risk, et SNMP põhisest UDP liiklusest saadaksegi näiteks kätte kogukonnanimi. See tähendab, et pealtkuulamine (*eavesdropping*) ja teesklus on kõige tõenäolisemad ohud, mis võivad rakenduda.

SNMPv1 täieliku turvalisuse puudumise tõttu kasutatakse seda enamasti agentide jälgimiseks. Tegelikult on enamikes seadmetes SNMP *set* protsessi kasutamine keelatud, sest ründajal on ülimalt lihtne teada saada vajalikud andmed seadme haldamise kohta. Isegi kui ründajal on olemas kogukonnanimi, siis saab ta pärida seadmest informatsiooni, kuid vähemalt ei ole tal võimalik seadme konfiguratsiooni muuta kuna *set* protsess on keelatud.

1.3.2 SNMP versioon 2

SNMP versiooni 2 (kasutatakse edaspidi ka lühendit SNMPv2) standardiseerimine ei olnud edukas. Standardiseerimise protsessi tulemuseks oli kolm omavahel ühildamatut

standardit: SNMPv2 osapoolte põhine (*party-based*), SNMPv2u ja SNMPv2*. SNMPv2-e määratlemine ja kavandamine algatati täiustamaks SNMP võimalusi ja samas oli mõningaseks prioriteediks ka turvalisuse tagamine. SNMPv2 osapoolte põhist ettepanekut ei võetud kasutusse ja seega jäi valida SNMPv2u ja SNMPv2* vahel, mis mõlemad on kasutajapõhiste mudelitega. Kahjuks tehti kompromiss ja ettepanek nimetusega SNMPv2c (mida tuntaksegi SNMPv2 all) standardiseeriti. Kahjuks sellepärast, et SNMPv2c-l on samamoodi olematu turvalisus nagu SNMPv1-l ja põhjuseks asjaolu, et turvalisuse mõttes ei tehtud mingisuguseid muudatusi SNMPv2c struktuuris võrreldes eelneva versiooniga. Endiselt kasutatakse turvalisuse tagamiseks kogukonnanimesid, mis on äärmiselt nõrk vahend informatsiooni kaitsmisel.

1.3.3 SNMP versioon 3

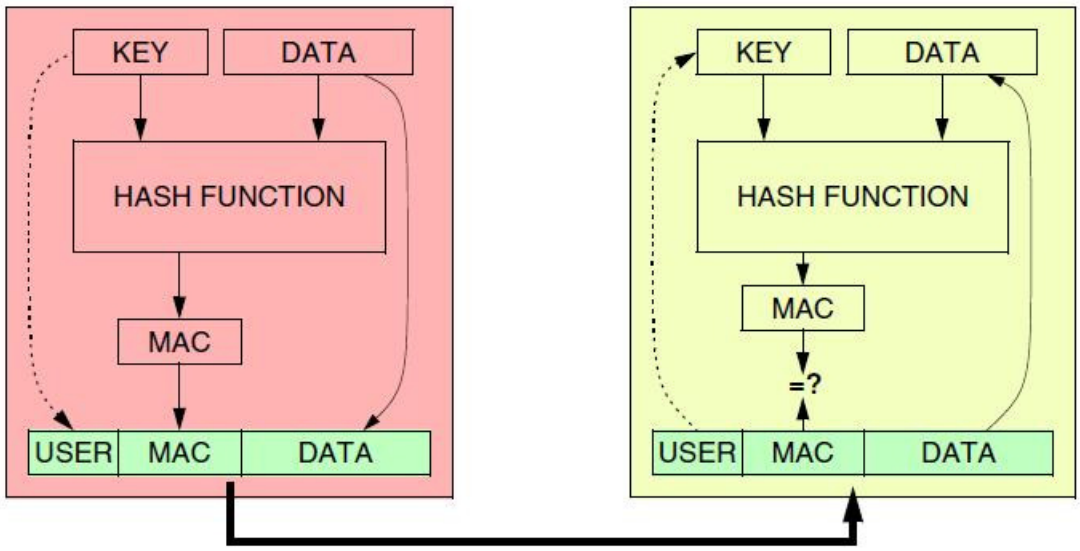
Kasutades SNMP versiooni 3 (edaspidi kasutuses ka lühend SNMPv3) on kasutajatel kindlus, et agentidelt kogutav haldusinformatsioon ei ole muudetud. Kasutajad võivad ka kindlad olla, et üle avaliku võrgu kogutav informatsioon on teada ainult selleks volitatud isikutele kuna rakendatakse krüpteerimist (kasutusel on DES ja AES krüpteerimisstandardid. Soovitav on kasutada AES-i, sest see tagab tugevama krüpteeringu võrreldes DES-iga). Lisaks võimaldab rühmapõhine haldusmudel kasutajatel ligi pääseda samale agendile erinevate õigustega ja erinevale haldusinformatsioonile.

SNMPv3 puhul kapseldatakse PDU, mis on versiooni 1 või 2 oma, SNMPv3 paketi sisse. NMS ja agent sisaldavad sõnumi töötluseks ühte SNMPv3 mootorit ehk programmi, mis haldab SNMP informatsiooni saatmist ja vastuvõtmist ning edastatavate sõnumite turvalisust tagavaid protsesse. Kui rakendus tahab saata PDU võrguseadmele, siis toimuvad järgmised tegevused: esmalt aktsepteerib mootor andmed, mida saadetakse SNMP rakenduskihilt; seejärel viib mootor läbi vajalikud turvalise funktsioonid; PDU kapseldatakse SNMPv3 sõnumi sisse ja sõnum saadetakse üle võrgu seadmele. Kui

seadme agent võtab sõnumi vastu, siis tehakse vajalikud autentimise ja dekrüpteerimise funktsioonid enne kui PDU edastatakse SNMP rakendusele.

Kasutajapõhine turvalisuse mudel (*User-Based Security Model* edaspidi USM) on mudel, mis teostabki turvalisuse teenuseid autentimise ja krüpteerimise jaoks. Selleks läheb vaja kahte erinevat salajast võtit, üks privaatsuse tagamiseks ehk krüpteerimisvõti või ka salajane võti- *privKey*- ja teine autentimise jaoks ehk autentimisvõti- *authKey*. Neid võtmeid ei hoita seadme MIB-is. Seega pole need ka kättesaadavad SNMP *get* ja *set* protsesside abil. USM abil kaitstakse haldusinformatsiooni teeskluse, informatsiooni muutmise, sõnumivoo muutmise ja avalikustamise ohtude vastu.

Saatja autentimiseks ja sõnumi tervikluse kontrollimiseks kasutab USM kahte erinevat autentimise protokolli (päringute juures on siis kasutusel üks kahest variandist), mis baseeruvad räsi põhisel sõnumi autentimiskoodil (*hash-based message authentication code* ehk edaspidi ka HMAC). HMAC-MD5-96 on protokoll, kus turvaline räsifunktsioon on MD5. HMAC-SHA-96 räsifunktsioon on SHA-1. Räsifunktsiooni sisendiks on nii sõnum kui ka salajane autentimisvõti *authKey*. Räsifunktsiooni väljundiks on sõnumi autentimiskood (*message authentication code* ehk edaspidi ka MAC), mis on oluliselt lühem kui sõnum ise. MAC lisatakse saadetavale sõnumile. Vastuvõtval poolel kasutatakse räsifunktsiooni sisendis *authKey*'d ja sõnumit ning arvutatakse MAC. Kui MAC ei ole sama, mis saatval poolel sõnumile lisatud sai, siis tervet sõnumit eiratakse. Kui MAC on sama, mis saatmisel, siis saab vastuvõttev pool kindel olla sõnumi tervikluses ja autentsuses. Sõnum ei saa olla muudetud selle ülekande ajal kuna ründaja peaks teadma salajast *authKey* võtit, et muuta haldusinformatsiooni andmeid. Samuti peaks ründaja õige MAC-i arvutamiseks teadma salajast võtit, kuid kuna see on teada ainult saatvale ja vastuvõtvale poolele, siis võib eeldada, et sõnum tuli autentselt osapoolelt. Autentimise ja tervikluse tagamise põhimõte on näidatud joonisel 7.

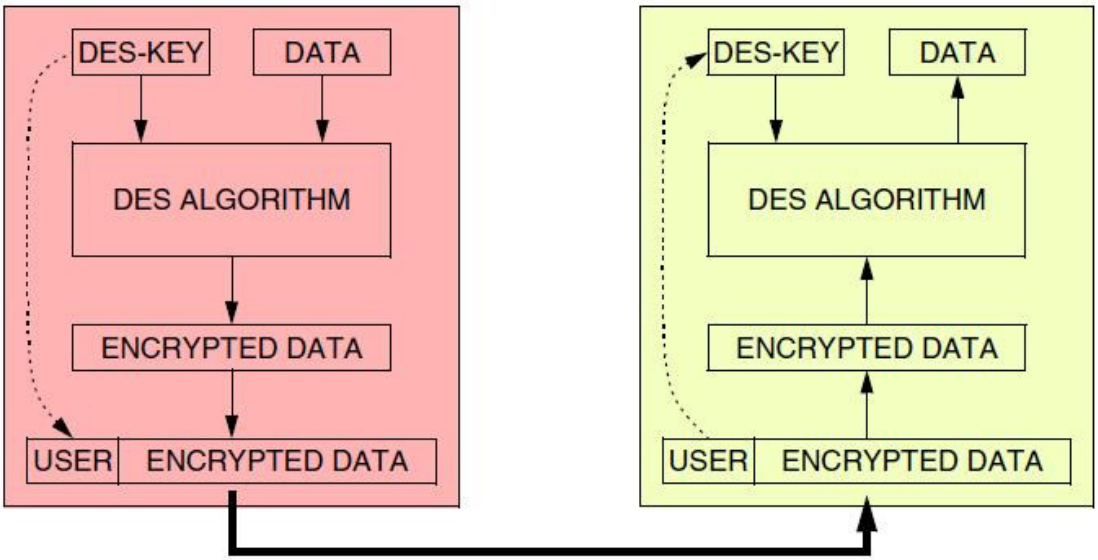


Joonis 7. SNMP versiooni 3 sõnumite autentimine [18]

Autentimisfunktsioon ei enneta sõnumi hilinemise või sõnumi uuesti saatmise ründeid kuna uuesti saadetud sõnumil ei ole tegelikult midagi tavatut. Selleks, et teha SNMPv3 turvaliseks taoliste rünnete vast, kasutab USM sõnumi õigeaegsuse tõendamise (*Timeliness Verification of Messages*) mehhanismi. SNMPv3 nõuab, et sõnumid oleks saabunud mõistlikus ajavahemikus. Sõnumi õigeaegsuse mehhanism baseerub kahele loendurile, mis on olemas igal SNMP agendil. Need on *snmpEngineBoots* ja *snmpEngineTime*. Kui agent installeeritakse, siis on mõlema loenduri väärtus null ja kui agent käivitatakse, siis *snmpEngineTime* suureneb iga sekundi möödudes ühe võrra. Kasutades keerulist sünkroniseerimise viisi, hoiab SNMP agent iga seadme umbkaudseid aja väärtusi. Need umbkaudsed aja väärtused pannakse kaasa iga saadetud sõnumiga (igas sõnumis sisaldub siis aja väärtus seadme kohta, millega NMS suhtleb). Teate vastu võtnud seade teeb kindlaks kas sissetulev sõnum on 150 sekundilise ajavahemiku sees ehk siis kontrollitakse, kas NMS-i saadetud teates sisalduv aeg on ligilähedane seadme agendi töötamise ajaga. Kui sõnum ei ole seadme agendi töötamise ajaga umbkaudu (150 sekundi vahemik) sama, siis sõnumit lihtsalt ei aktsepteerita.

Salastatuse jaoks kasutab USM andmekrüpteerimise standardit (*Data Encryption Standard* ehk edaspidi ka DES). DES algoritmi jaoks vajalik salajane võti on kasutajaga

seotud *privKey*. Sõnumite krüpteerimine on vabatahtlik ja nagu autentimisvõtmegi puhul hoitakse ka krüpteerimise võtit hallatavas masinas. Sõnumite krüpteerimise ja dekrüpteerimise põhimõte on sarnane autentimise protsessile, kuid vastuvõtval poolel on algoritmi sisendiks salajane võti ja krüpteeritud andmed ning väljundiks pole kontrollsõnum, nagu autentimise juures MAC, vaid saadetud PDU andmed. Kui krüpteeritud sõnum on saatmise ajal muudetud, siis dekrüpteerimisel saadud PDU sisu on tõenäoliselt arusaamatu ja sõnumit eiratakse. Salastatuse tagamise põhimõte on näidatud joonisel 8. Lisaks DES-ile on võimalik kasutada ka täiustatud krüpteerimisstandardit (*Advanced Encryption Standard* ehk AES).

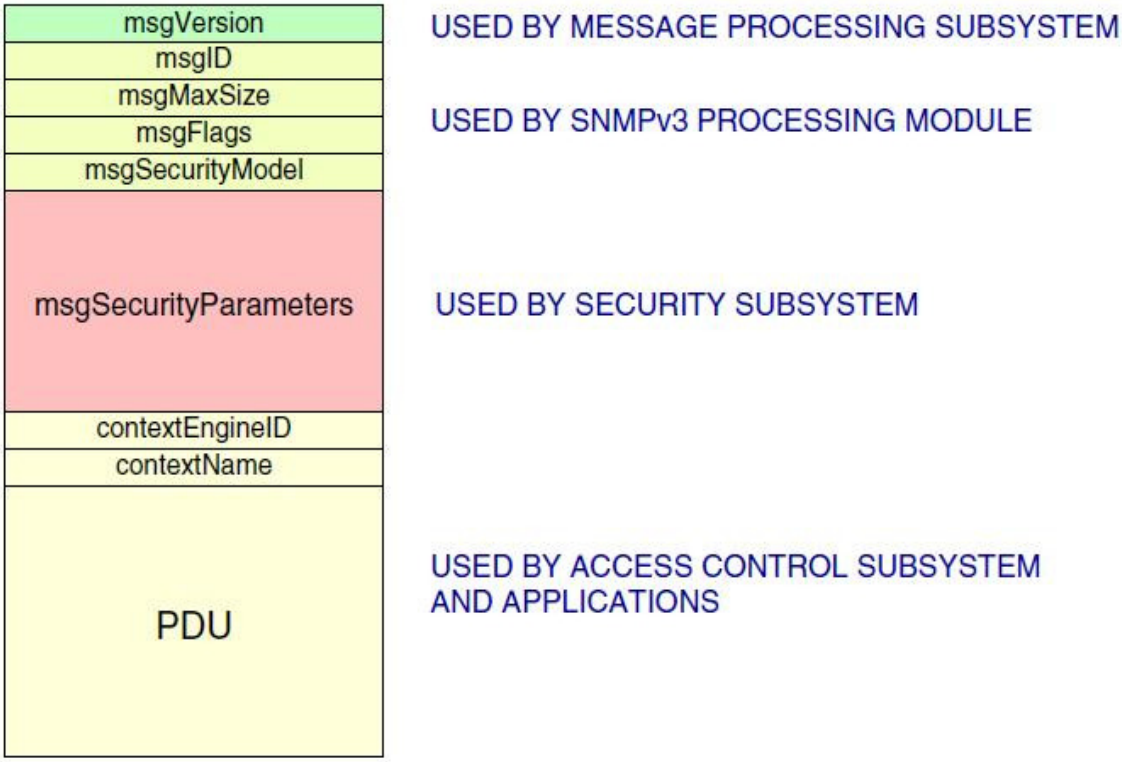


Joonis 8. SNMP versiooni 3 sõnumite krüpteerimine [18]

Ligipääsu reguleerimine (ehk teha kindlaks erinevate kasutajate õigusi vaadata või muuta seadme MIB-i) on turvalisuse funktsioon, mida teostatakse PDU tasemel ja selleks on erinevaid viise. SNMPv3 jaoks mõeldud ligipääsu reguleerimist kutsutakse vaatepõhiseks ligipääsu reguleerimiseks (*View-Based Access Control* ehk edaspidi ka VACM). VACM puhul määratakse kasutajate õigused rühma põhiselt. See on erinev USM-ist, kus kasutajate autentimine on individuaalne. VACM juures peab olema iga kasutaja määratud mõnda rühma ja rühmale antakse kindlad õigused süsteemis toimetamiseks. Näiteks administraatorite rühmale antakse täielikud õigused süsteemis tegutseda ehk vaadata

kõiki OID-sid, viia läbi nii *get* kui ka *set* käsud jne. Samas vaatlejate rühmale antakse ainult õigus süsteemist informatsiooni pärida mitte seadistada ja seda ka kindlale MIB-i osale.

Kuna SNMPv3-le on lisandunud oluliselt funktsionaalsust ja ta on tehtud ka märgatavalt turvalisemaks, siis lisandusid SNMPv3 sõnumile juurde mitmed väljad. Samas, nagu ka eelnevalt kirjutatud, PDU osa on sama, mis versiooni 1 või 2 oma ja PDU kapseldataksegi versiooni 3 sõnumi sisse. SNMPv3 sõnumi üldine struktuur on näha joonisel 9. *msgVersion* näitab SNMP versiooni. Järgnevad neli välja näitavad sõnumi haldamise parameetrid. *msgSecurityParameters* sisaldab omakorda mitmeid välju, mida läheb vaja turvalisuse tagamiseks. *contextEngineID, contextName* ja PDU on vajalikud ligipääsu reguleerimiseks ja haldusinformatsiooni andmete jaoks.



Joonis 9. SNMP versiooni 3 sõnumi üldine struktuur [18]

Paragrahvi „1.3 Turvalisus“ alusel on näha, et SNMP turvalisuse tagab ainult versioon 3. Versioonidel 1 ja 2 on nõrk autentimise funktsioon kogukonnanime kaudu, kuid igaüks, kes kuulab võrguliiklust pealt, saab kogukonnanime väga lihtsalt teada ja selle abil pärida infrastruktuuri seadmetest soovitud informatsiooni või siis muuta seadmete konfiguratsiooni. SNMPv3-l on aga turvalisuse funktsioonid, mis tagavad edastatavate SNMP sõnumite privaatsuse ja annavad kindluse, et teade tuleb õigest allikast ning seda pole ülekande käigus muudetud.

2. Monitooringuvahendid

Eelnevas peatükis on palju kirjutatud võrguhaldusjaamadest (NMS) ehk serveritest, kus peal töötab haldusülesandeid täitev tarkvara infrastruktuuri seadmete ja tarkvara seireks. Antud peatükis soovibki magistritöö autor kõigepealt välja tuua mõningad maailmas populaarsed tarkvarad (oma subjektiivse arvamuse alusel), mis täidavad võrguhaldusjaama ülesandeid. Seejärel on kirjeldatud nõuded, mis esitati Tallinna Tehnikaülikooli telekommunikatsiooni õppetooli õppejõudude poolt labori läbiviimiseks võrguhaldusjaama jaoks. Peatüki lõpus selgitab autor missugune NMS-i tarkvara labori jaoks valiti ja mis kujul see laborit läbi viivale õppejõule üle antakse.

2.1 Võrguhaldusjaamad

Erinevate võrguhaldusjaamade ülevaatlik tabel ja võrdlus on kätte saadav allikast [24]. Seda, kas antud informatsioon Wikipedia lehel 100% tõele vastab, ei saa magistritöö autor garanteerida, kuid siiski leiab sealt piisavalt informatsiooni võrguhaldusjaamade ja nende kasutamise võimaluste kohta. Kõik autori poolt kirjeldatavad NMS-id asuvad samuti Wikipedia lehel olevas tabelis.

2.1.1 Hewlett-Packard OpenView Operations

Materjal pärineb allikast [19, lk 20]

HP OpenView Operations (edaspidi kasutuses lühend OVO) kuulub Hewlett-Packardi OpenView tootekomplekti, mis koosneb võrgu ja süsteemi haldamise, andmesalvestuse, jõudluse kontrolli, auditi ja muudest taoliste funktsionaalsustega toodetest. OpenView lahendusega soovib Hewlett- Packard luua terviklahendust, millega on hallatav kogu IT

protsess ja sellele toetuvad erinevad ärilahendused. OVO lahenduse uus nimi on HP Operations Manager [49].

OVO ise on ehitatud Hewlett-Packardi varasemale tarkvarale Network Node Manager, mis olemuselt on pigem võrgu liikluse jälgimiseks mõeldud toode. OVO suunitlus seevastu on eelkõige masinate ja teenuste jälgimine, mille puhul paigaldatakse seadmele agent, mis kogub ja saadab informatsiooni kesksesse serverisse. Saadud informatsiooni põhjal genereeritakse raporteid ning graafikuid seadmete toimimisi kohta. Agente on võimalik seadistada ka täitma kindlaid tegevusi vigade esinemisel. OVO oskab jälgida ka logifaile filtreerides nendest välja vajaliku informatsiooni. Lisaks saab määrata ka ajastatud süsteemi kontrollide läbiviimise ja selle kontrolli tulemusest sõltuvalt (kas kontrolli tulemus oli edukas või mitte) käivitada ettemääratud tegevused.

Kõik masinates toimuvad intsidendid seotakse OVO *Alert*idega, mis saadetakse edasi haldusserverisse. *Alert*idega tegeleb selleks määratud administraator ja samas on administraatoril võimalik probleem ka tegelemiseks edasi suunata mõnele teisele administraatorile.

Hewlett-Packard OpenView Operations on võrguhalduslahendus, mis annab kiire ülevaate infrastruktuurist ja mida on administraatoril masinate haldamiseks lihtne ning mugav kasutada. OVO lahendust saab juurutada Windows, Linux, HP-UX ja Solaris operatsioonisüsteemidel [50]. OVO on tasuline võrguhalduslahendus.

2.1.2 ServersCheck Monitoring Software

ServersCheck on mõeldud võrguseadmete, serverite ja rakenduste seireks, vigadest raporteerimiseks ja vigadest teavitamiseks. Jälgitavate seadmete haldamine ja hetkeseis on nähtav veebilehitseja kaudu. [20, lk 10]

ServersCheck suudab läbi viia üle 60 erineva kontrolli, sealhulgas jälgida seadmeid kasutades ICMP-d, kontrollida kõiki TCP porte ja e-posti protokolle, pärida SNMP andmeid, jälgida Windows, Linux ja Unix süsteemide koormusnäitajaid ja protsesse. Infrastruktuuri vigade korral on ServersCheck võimeline kasutajaid teavitama mitmel erineval viisil sealhulgas e-posti, SMS-i ja MSN-i kaudu. Teateid saab saata ühele või mitmele kasutajale korraga. Lisaks näitab ServersCheck statistikat, raporteid ja graafikuid jälgitavate objektide kohta. [51]

ServersChecki positiivseks küljeks on mitme erineva keele tugi, sealhulgas on toetatud ka eesti keel. Toodet saab osta kahel kujul: autonoomse seadmena, kuhu on juba tarkvara installeeritud või tarkvarana, mis on installeeritav Windows operatsioonisüsteemile. ServersCheck on tasuline lahendus. [51]

2.1.3 Zabbix

Materjal pärineb allikast [52].

Zabbix suudab jälgida mitmeid infrastruktuuri parameetreid ning võrguseadmete ja arvutite probleemidele kiireks reageerimiseks on võimalik määrata kasutajad, keda juhtunust teavitatakse. Jälgitavate seadmete kohta informatsioon ja seadistused on kätte saadavad veebipõhise kasutajaliidese kaudu.

Zabbix on võimeline monitoorima infrastruktuuri erinevaid komponente alates teenustest nagu turvaline kaugligipääsu teenus ning protokollidest ICMP, SNMP ja lõpetates koormuste jälgimisega ning erinevate operatsioonisüsteemide protsesside ja teenuste kontrollimisega. Süsteemide toimimisest saab hea ülevaate graafikute, raportite ja statistika abil. Lisaks on Zabbixil auditeerimise võimalus, mis annab informatsiooni konfiguratsiooni muudatuste kohta.

Zabbixi võrguhaldusjaama tarkvara on võimalik paigaldada Linux, Solaris, HP-UX, AIX, Free BSD, Open BSD ja Mac OS X operatsioonisüsteemidele. Zabbix on kasutamiseks tasuta, GPL litsentsi alusel.

2.1.4 OpenNMS

OpenNMS on võrguhalduse lahendus, mis suudab automaatselt üles otsida võrgus olevad seadmed ja kasutatavad teenused (kuid loomulikult saab neid määrata ka administraator ise) ning monitoorida nende korrektset toimimist ja teavitab infrastruktuuri probleemide korral määratuid kasutajaid näiteks e-posti või SMS-i teel. OpenNMS jaoks on haldus on veebilehitseja põhine. [53]

OpenNMS suudab jälgida väga paljusid erinevaid teenuseid/ teenuseid tagavaid protokolle, sealhulgas DHCP, DNS, NTP, SSH, kontrollida andmebaaside (näiteks Oracle, Postgres, MySQL) tööd, koguda seadmete ressursi tarbimise infot kasutades erinevaid meetodeid, millest üks on kindlasti protokoll SNMP, monitoorida erinevate operatsioonisüsteemide protsesse ja teenuseid jne. Lisaks on võimalik teha detailseid raporteid teenuste ning seadmete toimise kohta ja näha graafikuid infrastruktuuri ressursside kasutuse kohta. [54]

OpenNMS võrguhalduslahendust saab juurutada Windows, Mac OS X ja Solaris operatsioonisüsteemidel ning erinevatel Linuxi distributsioonidel. [55] OpenNMS on tasuta tarkvara [56].

2.1.5 Nagios

Materjal pärineb allikast [21].

Nagios on võrguhaldusjaama tarkvara, mis võimaldab jälgida teenuste, serverite ja võrguseadmete kättesaadavust. Infrastruktuuri probleemide korral saadab Nagios välja

teavitusi määratud kasutajatele. Teavituse viise on erinevaid, kuid nende hulka kuulub kindlasti e-posti ja SMS-i kasutamise võimalus. Nagiose haldus toimub veebipõhise kasutajaliidese kaudu, kuid jälgitavad seadmed ja teenused tuleb eelnevalt ära määrata kindlas konfiguratsioonifailis.

Nagiosele on kirjutatud terve hulk pistikprogramme (*plugin*), mis ongi Nagiose töö aluseks. Pistikprogrammide abil päritakse infrastruktuurist kõikvõimalikke erinevaid teenuseid (näiteks SMTP, IMAP, HTTP, FTP, DNS), seadistusi, jälgitakse serveri "sisemist" infot, nagu koormus, kettamaht, protsesside arv, kontrollitakse seadme ülevalolekut jne. Sellisteks pistikprogrammideks on näiteks Cisco seadmete staatuste kogumine, HTTP teenuse töötamine kindlas serveris jms. Nagioses on olemas statistika ja raportite kuvamise võimalus, kuid lisaks on Nagiost võimalik siduda mitmete muude tarkvaradega, näiteks NagiosGrapher tarkvaragaga Nagiosest kogutud informatsiooni raporteerimiseks graafikute kujul (lähemalt juttu paragrahvis „5.3 Raporteerimistarkvara“) või Syslog-ng tarkvara Nagiose logide haldamiseks.

Nagiose juurutamiseks on arvutile ainukesteks nõueteks Linuxi distributsiooni ja C kompilaatori olemasolu. Nagiost on võimalik paigaldada ka Windowsi operatsioonisüsteemiga arvuti peale, kuid see on alles beeta testimisel (57). Nagios on kasutamiseks tasuta, GPL litsentsi alusel.

2.2 Ülikooli nõuded laborile

Magistritöö autor arutas labori jaoks loodava võrguhaldussüsteemi üle enne töö tegema hakkamist telekommunikatsiooni õppetooli õppejõududega. Üldiselt anti autorile vabad käed luua oma parema äranägemise järgi SNMP labor, mis teeks üliõpilastele selgeks SNMP ja selle toimimise ning annaks praktilise kogemuse võrguhaldusjaamaga töötamisel. Samas esitati kaks olulist nõuet laboris kasutatava süsteemi kohta.

Praeguse Andmeside võrguhalduse labori juures on üks negatiivne külg ja selleks on kasutatav tasuline tarkvara InterMapper. Kui üliõpilane soovib ise hetkel laboris kasutatavat tarkvara testida või seda ise oma töökeskkonnas juurutada, siis saab ta seda teha piiratud aja jooksul (2 nädalat) [46]. Pärast seda tuleb osta tarkvara kasutamiseks litsents. Litsents, millega saab jälgida kuni 25 seadet ja nende võrguliiklust, maksab peaaegu 620 eurot [22].Seetõttu pandigi õppejõudude poolt töö tegemiseks peale piirang, et laboris kasutatav tarkvara peab olema tasuta, et seda saaks nii ülikool kui ka (soovi korral) üliõpilased igal ajal kasutada ja nii kaua kui soovivad.

Telekommunikatsiooni õppejõudude poolt oli lisaks veel nõue, et kogu laboris kasutatav infosüsteem oleks vabavaraline ja käivitatav VMware virtualiseerimisvahenditega. Nõue tulenes asjaolust, et vajadusel saaks laboris kasutatavale NMS-ile lisaprogramme juurde installeerida ja süsteemi edasi arendada. Virtualiseerimise soov oli asjaolus, et võrguhaldusjaama töötaks olemasoleva riistvara peal ehk siis poleks vaja juurde soetada eraldi serverit magistritöö autori labori jaoks. Samuti saab (soovi korral) üliõpilastele testimiseks või ka reaalseks kasutamiseks anda valmis võrguhaldusjaama, kus oleks vaja ainult muuta võrguparameetrid ja lisada mõningad seadistused, et jälgida infrastruktuuri (olgu need siis üliõpilase kodus olevad seadmed või terve ettevõtte seadmete/ arvutite park).

2.3 Laboris kasutatav võrguhaldusjaam

Paragrahvi „2.1 Võrguhaldusjaamad“ all tutvustas töö autor maailmas populaarseid NMS-e. Käesolevas paragrahvis ei ole aga eesmärk hakata neid võrguhaldusjaamu võrdlema ja kaaluma nende negatiivseid ja positiivseid külgi. Selle töö on juba paljud inimesed oma kirjutistes ja uurimistöödes ära teinud (näiteks ülalmainitud Wikipedia lehel on võimalik hea ülevaade saada. Samuti leiab mõningatest diplomitöödest erinevate NMS-ide kohta võrdluse [19] [20]). Võrguhaldusjaamade paragrahvi eesmärk oli informatiivselt tutvustada neid NMS-e, mis on laialdaselt kasutuses. Autori arvates on paragrahvis „2.1 Võrguhaldusjaamad“ kirjeldatud süsteemid ennast üldiselt tõestanud ja

näidanud tarkvara heast küljest, mis tähendab, et neid süsteeme võib julgelt kasutusele võtta infrastruktuuri monitooringusüsteemina. Kindlasti tutvustatakse labori käigus ühte võrguhaldusjaama lähemalt, et tudengitel oleks ülevaade reaalselt toimivast võrguseire lahendusest. Loomulikult oleks hea tutvustada tudengitele kõiki paragrahvis „2.1 Võrguhaldusjaamad“ nimetatud võrguhaldusjaamu, kuid labori ajalise piirangu tõttu ei ole see võimalik.

Magistritöö tulemusena tehtavas laboris kasutatakse võrguhaldusjaamana Nagiost. Selleks on põhjusi mitmeid, kuid eelkõige kallutas valiku just Nagiose suunas autori kokkupuude nimetatud NMS-iga (valik oli ka tegelikult kolme vahel, sest esimesed kaks tutvustatud võrguhaldusjaama on tasulised ja seega ei täida õppejõudude esitatud nõudeid). Samas tuleb tõdeda, et Nagios ei olnud suvaline eelistus. Nii palju kui autor on seda võrguhaldusjaama kasutanud, saab tõdeda, et Nagios on haldaja (ehk siis laboris üliõpilase) jaoks kergesti jälgitav ja arusaadav. Puuduseks võib pidada ehk seda, et peab tegema liialt palju käsitsi tööd monitooringufaili loomisel ja vaatama, et seadmete ning päringute määramisel vigu ei tekiks. Samas on see üliõpilase jaoks pigem kasulik kui ta saab ise kirjutada valmis monitooringufaili ja rida-rea haaval uurida, kuidas süsteem päringuid teostab ja mida selleks kasutab. Nagiose head tööd hindavad ka Eestis nii mõnedki erafirmad [23] kui ka avaliku sektori esindajad.

Labori jaoks installeeris autor VMware virtuaalmasinale vabavaralise operatsioonisüsteemi/ Linuxi distributsiooni 32-bitise Fedora *release* 12 (Constantine). Virtuaalmasinale on antud muutmälu 1 GB, kõvakettamahtu 10 GB-i ja kasutuses on 32-bitine arhitektuur. Lisaks paigaldas autor pärast Fedora installeerimist ka veebiserveri (Nagiose puhul vajalik üldise ülevaate saamiseks hetkeseisust võrgus) ja failiserveri (mugavam kasutajatel monitooringuseadistusi hallata) tarkvara. Seejärel paigaldas autor järgmised komponendid, et saakski hakata infrastruktuuri jälgima:

Nagios versioon 3.2.1 – võrguhaldusjaama tarkvara. Allalaadimislink internetis on [25].

Nagios-Plugins versioon 1.4.14 – pistikprogrammid, mis tegelevad päringute teostamisega. Allalaadimislink internetis on [26].

Net-SNMP versioon 5.4.2.1 – SNMP päringute teostamiseks ja *trap*'ide vastuvõtmiseks vajalik pakett. Allalaadimislink internetis on [27].
SNMPTT versioon 1.3 – vastuvõetud *trap*'ide töötlemiseks ja Nagiosele edasisaatmiseks vajalik pakett. Allalaadimislink internetis on [28].

Eelnimetatud komponentide paigaldamise järel on Nagios valmis teostama võrgumonitooringut ning vastu võtma *trap*'e. Nagiose dokumentatsioon ning paigaldusjuhend on internetist kättesaadav leheküljelt [29]. Samuti peab tegema teatud seadistusi, et seadmetelt saadetud *trap*'e saaks vastu võtta ja need saadetakse Nagiosele õigel kujul edasi. Selle jaoks on olemas internetis väga hea juhend leheküljel [30].

Seadistatud ja kõikide vajalike komponentidega virtuaalmasin antakse üle labori juhendavale õppejõule. Lisaks antakse üle ka seadistuste dokumentatsioon.

3. Maailma ülikoolides kasutatavad SNMP laborid

Enne SNMP labori loomist Andmeside kursuse jaoks uurib magistritöö autor missuguseid ülesandeid tehakse erinevates maailma ülikoolides võrguhalduse valdkonnas. Kuna paljudes infotehnoloogiat õpetavates kõrgkoolides on SNMP laborid kasutusel, siis valib antud töö autor mõned nendest välja (valik on tehtud magistritöö autori subjektiivse otsuse põhjal arvestades labori keerukust ja kasutusvõimalusi koostatavas laboritöös) ja kirjeldab, mis ülesandeid tuleb üliõpilastel lahendada. Samuti analüüsitakse mida ja miks võiks erinevatest maailma ülikoolide välja töötatud või ülikoolides kasutatavatest SNMP laboritest kasutada ka Andmeside võrguhalduse laboris.

3.1 California State University, Fresno

Ülikooli laborid on kättesaadavad allikatest [31] ja [32]. Antud laborite eesmärk on õpetada SNMP agentide paigaldamist ja seadistamist Windowsi keskkonnas. Ülikooli laborid baseeruvad Cisco CNAP IT Essentials II - Network Operating Systems 2.0 programmile.

Esimeses laboris antakse ülevaade kuidas ja mille alt saab Windowsi operatsioonisüsteemiga arvutisse installeerida SNMP agendi. Lisaks antakse ülevaade, kuidas seadistada arvuti asukoha ja kontakti kirjeldus ning missuguseid võrguteenuseid saab SNMP agendi käest pärida ja agendi abil määrata.

Teises laboris antakse juba konkreetsemalt juhised turvalisuse ja *trap*'ide seadistamiseks. Õpetatakse kuidas ja kuhu tuleb lisada kogukonnanimi päringute autentimiseks ning kas SNMP päringuid seadistatavasse arvutisse lubatakse teha kindlatest masinatest või kõikidest võrgus olevatest masinatest. Lisaks antakse ülevaade kuidas seadistada saadetavate *trap*'ide sihtkoht.

Käesolevas paragrahvis analüüsitavad laborid on suhteliselt lihtsad, kuid annavad siiski ülevaate väga olulisest asjast, milleks on Windowsi operatsioonisüsteemiga arvutitele SNMP agentide installeerimine ja seadistamine. Kuna Windowsi operatsioonisüsteemiga arvutid on maailmas väga laialdaselt kasutuses, siis on kindlasti nii mõnelgi võrguhaldusega tegeleval infotehnoloogia- või telekommunikatsioonispetsialistil huvi tööjaamu või servereid SNMP abil jälgida. Vaikimisi ei ole SNMP agendid Windowsi keskkonnaga arvutitesse installeeritud [61], seetõttu ongi esimene laboritest asjalik, sest annab juhised SNMP agendi paigaldamiseks ja teine labor näitab juba konkreetsemaid seadistusi Windowsi arvuti jaoks kogukonnanime määramiseks ja *trap*'ide saatmise seadistamiseks. Nimetatud põhjuste tõttu võiks olla California State University, Fresno laborid ka osaks magistritöö autori laboris või siis vähemalt labori juhendis peaks selle kohta olema viide, et üliõpilased teaksid kuidas SNMP agent Windowsi keskkonda paigaldada.

3.2 Ohio University

Ülikooli labor on kätte saadav internetist leheküljelt [33]. Labori eesmärk on uurida TCP ühenduste läbilaskevõimet TCP seadistuste muutmisel. Samuti on eesmärgiks uurida SNMP abil marsruuteri seadistusi ja liideste võrguliikluse statistikat ning võrrelda SNMP abil kogutud informatsiooni marsruuteri enda kasutajaliidesest väljastatud informatsiooniga.

Labori põhirõhk on TCP ühenduste läbilaskevõime uurimisel erinevate TCP seadistuste korral- näiteks uuritakse TCP akna suuruse muutmist erinevatel operatsioonisüsteemidel ja kuidas see mõjutab saadetavat infohulka. Testid teostatakse programmi *iperf* abil. Ühe osana laborist on kaasatud ka SNMP uurimine, mille abil päritakse esmalt marsruuteri nime kahel erineval kujul. Päringute vastused ühel juhul on inimloetaval kujul ehk tekstiliselt ja teisel juhul OID-na ehk numbrilisel kujul. Lisaks uuritakse SNMP abil marsruuteri liideseid läbinud võrguliikluse hulka erinevate SNMP protsesside abil ning hiljem võrreldakse saadud tulemusi Cisco marsruuteri kasutajaliidesest saadud

informatsiooniga. SNMP liikluse uurimiseks ja pakettide püüdmiseks kasutatakse ka võrguprotokollide analüsaatorprogrammi Wireshark.

Ohio University labor on SNMP poole pealt vaadates asjalik. SNMP-d kasutatakse küll suhteliselt vähe terve labori jooksul, kuid see annab marsruuteri võrguliideste kohta üliõpilasele väga palju informatsiooni. Lisaks kasutatakse selles laboris *get* (selleks *snmpget* käsk) ja *getnext* (selleks *snmpwalk* käsk) protsesse, mis on SNMP päringute teostamisel põhilised seadmest informatsiooni kogumise vahendid (SNMP protsessidega seotud käske tutvustatakse lähemalt punkti „4.2.1 Töö käik“ alapunktides). Tutvustatakse ka *getbulk* (selleks *snmpbulkget* ja *snmpbulkwalk* käsud) protsessi. SNMP pakettide uurimiseks on laboris kasutusel Wireshark tarkvara, mis näitab üliõpilastele juba konkreetselt, mis informatsiooni ja mis kujul SNMP päringute ja vastuste paketid sisaldavad. Kuna eelmainitud protsessid ja neid protsesse rakendavad käsud on SNMP tundma õppimiseks väga olulised, siis kasutatakse kindlasti ka Andmeside laboris Ohio University labori näiteid. Samuti laseb magistritöö autor kindlasti üliõpilastel pakettide analüüsimiseks kasutada Wireshark programmi kuna see annab ülevaate kogu informatsioonist SNMP päringute teostamisel ja päringutele vastuste saamisel.

3.3 Boston University

Ülikooli SNMP labor on leheküljel [34]. Labori eesmärgiks on tutvustada SNMP-d, vaadelda MIB failide sisu, õpetada NMS-i abil pärima seadmetest informatsiooni ja näidata kuidas saab kätte võrguseadmete seadistused kasutades SNMP-d.

Labor on detailne ja nõuab üliõpilaselt enne labori tegema asumist iseseisvat tööd, et SNMP alused ja ülevaade oleks läbitud. Samuti nõuab labor eelnevat tutvumist Net-SNMP tarkvarapaketiga, et ülesannete lahendamise ajal oleks üliõpilasel selge, missugune programm viib läbi kindla SNMP protsessi. Kogu labor koosneb neljast osast. Iga osa hõlmab mingit SNMP komponenti või SNMP rakendamist reaalelulises situatsioonis. Erinevate osade alguses on seletatud lahti, mis SNMP-d puudutava asjaga

on tegemist, seejärel peab üliõpilane viima läbi ettenähtud tegevused ja vastama laboris esitatud küsimustele.

Labori esimeses osas antakse lühike ülevaade MIB-idest ning OID-dest ja seletatakse MIB-ide süntaksi. Samuti on illustreeritud MIB-i objektipuu. Üliõpilase ülesandeks on tutvuda mõningate MIB-2 alampuu haldusinfobaasidega ja seal sisalduvate OID objektidega. Liskas peab üliõpilane otsima välja kindlate OID objektide definitsioonid ja kasutades *snmptranslate* käsku Linuxi masinas vaatama nende objektide kuju nii numbrilisel kui ka tekstilisel kujul. Labori teises osas tutvustatakse esmalt SNMP agente ning võrguhaldusjaamu ja nende vahelist suhtlust. Kirjeldatakse lühidalt ka SNMP põhilisi protsesse- *get*, *getnext*, *set* ja *trap*- ning nende toimimist. Üliõpilane peab pärima Linuxi masina SNMP agendist ja Cisco marsruuterilt erinevaid OID-sid ja võrdlema erinevate SNMP protsesside tööpõhimõtteid. Labori kolmas osa kirjeldab lühidalt SNMP *trap*'i toimimist. Cisco marsruuteris seadistatakse *trap*'ide saatmine ja määratakse saadetavate *trap*'ide sihtkoht ehk võrguhaldusjaam. Seejärel saadetakse Cisco marsruuterist kindlad *trap* teated. Üliõpilane näeb neid teateid NMS-i terminali aknas ja salvestab saadud tulemused, et neid uurida ja nende põhjal vastata laboris esitatud küsimustele. Labori neljas osa laseb üliõpilastel teha SNMP abil reaalsed võrguhalduse ülesanded. Esiteks peab üliõpilane SNMP abil pärima kindlast arvutist TCP ühenduste informatsiooni, mille tagajärjel on võimalik kindlaks määrata kui palju kasutajaid on selle kindla arvutiga ühenduse loonud. Teiseks peab üliõpilane määrama SNMP abil kahe arvuti vahelise võrguliikluse teekonna. Viimane labori osa näitabki üliõpilastele mõningad viisid kuidas saab SNMP-d kasutades reaalselt võrguseadmeid hallata ja võrgus toimuva kohta vajalikku informatsiooni.

Boston University SNMP labor on kahtlemata hästi üles ehitatud ja sisaldab kogu vajalikku informatsiooni, et üliõpilane saaks ülevaate ja arusaamad SNMP alustest (seda muidugi eelnevalt mõningase iseseisva töö läbimisega). Väga hea on selle labori juures ka asjaolu, et iga osa alguses on lühidalt lahti seletatud, millega tegemist tuleb ja miks see vajalik on. Lisaks on äärmiselt positiivne, et labori viimane osa laseb kasutajal läbi teha reaalelulised näited andmaks üliõpilasele natukenegi aimu miks üldse SNMP-d vaja läheb

ja kuidas seda oma igapäevatöös oleks võimalik rakendada. Magistritöö autor arvestab Boston University SNMP labori ideedega ja lisab Andmeside kursuse laborisse kindlasti Boston University labori neljanda osa põhimõttel mõne ülesande, mis kajastaks üliõpilasele reaalset töös vajalikku SNMP kasutust.

3.4 ISSNSM - International Summer School on Network and Service Management

Suveülikooli labor on kätte saadav allikast [35]. Laborit viib läbi Aiko Pras, kes on professor Hollandis Twente Ülikooli Arvutiteaduste Instituudis. Aiko Pras on tunnustatud teadlane võrguhalduse valdkonnas [62]. Tema suveülikoolis juhendatava labori eesmärgiks on anda SNMP-st ja selle komponentidest põhjalikum ülevaade, et doktoriõppe tudengid oleksid teadlikud üldistest vigadest, mida tehakse SNMP-ga seotud publikatsioonides. Siinkohal tulekski toonitada, et labor on mõeldud eelkõige doktoriõppe tudengite jaoks. Selle suveülikooli labori tegemiseks peaksid olema head teadmised SNMP-st või tuleks teatud hulk materjali enne harjutuste tegemist läbi töötada, et laborit üldse oskaks tudeng teha.

Labor koosneb neljast osast. Esimeses osas tuleb etteantud objektide alused tudengil ise valmis kirjutada MIB moodul ja hiljem kontrollida kas mooduli süntaks on õige. Teises osas peab otsima õigetest MIB failidest informatsiooni ja pärima õigeid OID-sid, et vastata harjutuses esitatud küsimustele. Kolmandas harjutuses päritakse OID-sid kasutades erinevaid SNMP versioone. Labori neljas osa laseb dekodeerida kinni püütud SNMP sõnumit, et sealt kätte saada näiteks kogukonna nimi ja OID-d.

Antud suveülikooli labori läbimine annaks väga heal tasemel teadmised SNMP-st, sest õpitakse ise kirjutama MIB moodulit ja laboris esitatud küsimustele vastamine nõuab SNMP erinevate versioonide ning MIB failide pikemat analüüsimist. Nende ülesannete täitmine aga eeldab juba eelnevalt suhteliselt põhjalikke teadmisi SNMP-st. Samas Andmeside kursuse jaoks tehtav labor ei oota üliõpilastelt eelnevalt mingeid kokkupuuteid SNMP-ga. Seega ei saa selle suveülikooli laborist otseselt midagi

rakendada magistritöö käigus loodava labori jaoks. Ainuke võimalus oleks ehk kasutada suveülikooli laborist teise ja kolmanda osa põhimõtteid, et tutvustada üliõpilastele SNMP versioonide erinevusi ja lasta uurida MIB failide sisu.

3.5 The University of Texas at San Antonio

Ülikooli labor asub allika [36] all. Labori eesmärgiks on anda ülevaade SNMP kasutamisest võrguhalduse toimingute jaoks, et kasutajad teaks kuidas jälgida arvuteid (antud laboris Linuxi näitel) ja Cisco marsruutereid SNMP abil, teaksid mis on võrguhaldusjaam ja milleks seda kasutatakse, saaksid aru kogukonnanime funktsioonist ning oskaksid kasutada *set* ja *trap* protsessi.

Ülikooli labor eeldab mõnigast materjalidega tutvumist Linuxis kasutatavate SNMP käskude koht- *snmpget*, *snmptranslate*, *snmpwalk* ja *snmpset* -, kuid selle labori juures saab hakkama ka ilma eelenevalt materjalidega tutvumata. Labori käigus paigaldatakse võrguhaldusjaamale Net-SNMP pakett, et saaks teha päringuid jälgitavate arvuti kohta. NMS-is käivitatakse ka *snmptrapd* program *trap*'ide vastuvõtmiseks. Samuti käivitatakse *snmpd* programm Linuxi tööjaamdes, et SNMP agent vastaks NMS-i poolt tehtavatele päringutele. Seadistatakse ka Cisco marsruuterid, et neid saaks SNMP kaudu hallata. Seejärel uuritakse *snmpget* ja *snmpwalk* käskudega Cisco marsruuteri MIB-i struktuuri ja mõningate OID-de väärtusi. Lõpuks peab üliõpilane *snmpset* käsuga muutma marsruuteri süsteemi nime ja lülitama välja ühe marsruuteri võrguliidese, et NMS-ile saadetaks *trap*.

The University of Texas at San Antonio SNMP harjutused on võrreldes eelpool kirjeldatud laboritega kõige ülevaatlikum ja lihtsamini jälgitav. Konkreetselt näidatakse ära, mis käsk täidab mingit ülesannet ja paari toetava küsimuse abil tehakse üliõpilasele arusaadavaks mida antud harjutus annab. Kõik käesoleva paragrahvi laboris kasutatavad SNMP käsud leiavad rakendust ka Andmeside kursusele tehtavas laboris kuna nende abil saabki üliõpilastele anda algteadmised SNMP-st ja näidata, mida kasulikku võrguhalduse protokoll abil on võimalik teha.

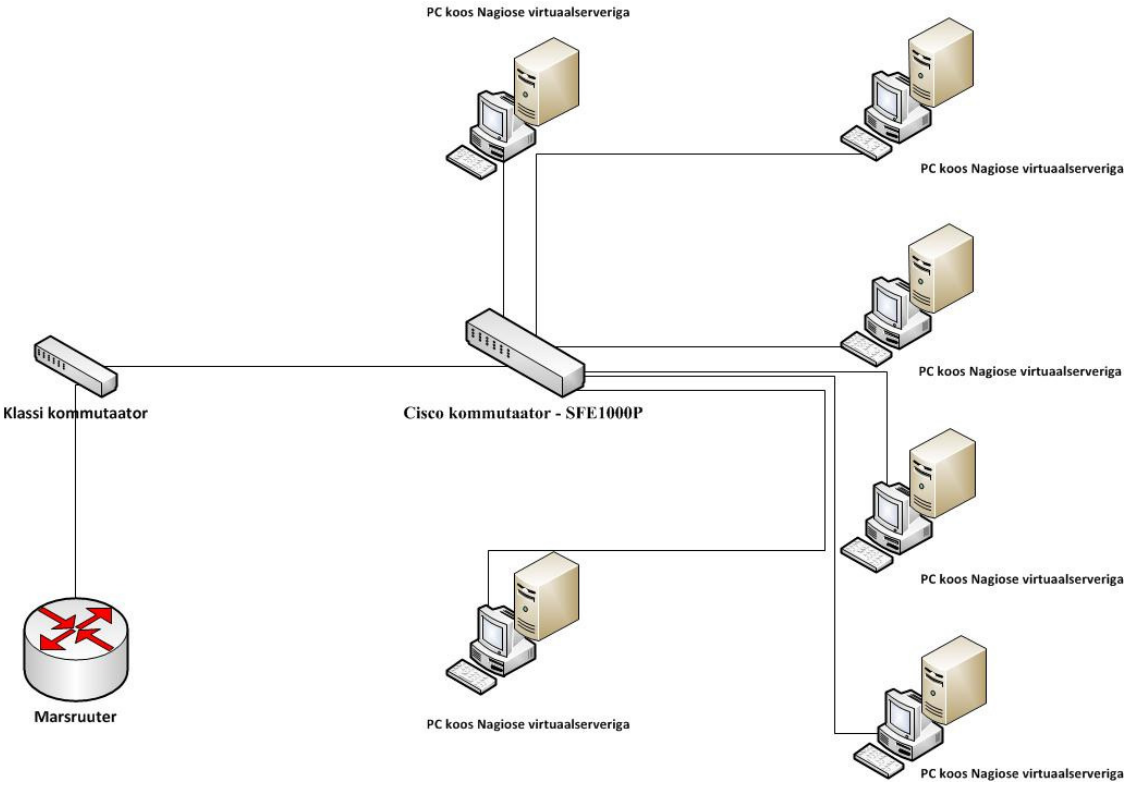
4. Andmeside kursuse labor

Antud peatükk kirjeldab juba konkreetselt, missugusena magistritöö autor näeb magistriõppe kursuse võrguhalduse laborit ja mis harjutusi tuleb üliõpilastel teha, et labor edukalt sooritada.

SNMP võrguhalduse labor tervikuna koosneb kahest eraldiseisvast osast, millest üks on üliõpilase jaoks teoreetiline ja teine praktiline. Teoreetiline osa on lühem ja kestab umbes 20 minutit. Selle käigus teeb õppejõud tudengitele ettekande, millega annab lühikese ülevaate SNMP-st, võrguhaldusjaamadest (rõhk Nagiosel), labori topoloogiast ja selgitab, mis harjutusi tudengid tegema peavad. Praktiline osa on pikem ja kestab umbes tund ja kümme minutit. Kogu labori pikkuseks saab olla maksimaalselt poolteist tundi. Praktilise osa puhul tuleb tudengitel teha lühike laboritöö ettevalmistus, et saaks alustada laborit, seejärel läbida harjutused, mis on ette nähtud ning lõpuks koostada laboris tehtu kohta aruanne, kus on kirjas üliõpilase tegevused ja vastused labori juhendis esitatud küsimustele. Aruanne esitatakse õppejõule elektroonilisel kujul veebilehena. Pärast aruande esitamist peab iga tudeng ka oma tehtud tööd kaitsma, et labor arvestatud saaks. Praktilise osa tegemiseks antakse üliõpilasele ette laboritöö juhend, mis sisaldab kogu vajalikku informatsiooni labori ja harjutuste kohta. Lisaks on laboritöö juhendi lõpus lühike õpetus programmi Wireshark kohta- kuna seda läheb pakettide püüdmisel vaja- ja lingid, kust üliõpilane leiab täiendavat informatsiooni SNMP kohta, kui peaks laboris või küsimuste vastamisel probleeme tekkima. Laboritöö juhend pannakse üles Andmeside kursuse kodulehele [37]. Labori teoreetilise ja praktilise osa ülevaade on järgmiste paragrahvidena antud peatükis kirjeldatud. Teoreetiline osa valmis kujul ehk ettekanne on toodud lisas 1 ja praktilise osa tegemiseks vajalik laboritöö juhend on toodud lisas 2.

Labori teoreetilise osa läbiviimiseks on õppejõul vaja projektorit ja arvutit, et teha üliõpilastele ettekanne. Praktilise osa tegemiseks on vaja arvutiklassi, kus arvutitel oleks piisavalt ressurssi, et käivitada ja töös hoida Nagiose virtuaalmasinat. Selliseid arvuteid

on Tallinna Tehnikaülikoolis võrguhaldust läbi viidavas klassis kuus tükki. Virtuaalmasina käivitamiseks on igas arvutis vajalik programmi VMware Player olemasolu. Nimetatud programm on tasuta kätte saadav allikast [38]. Eelnevalt tuleb siiski ennast registreerida. Lisaks sellele peab pakettide püüdmiseks olema igas arvutis ka programm Wireshark, mis on samuti tasuta ja kätte saadav allikast [39]. Labori mitmekülgsemaks tegemiseks oleks hea kasutada ka mõnda võrguseadet, mida SNMP abil hallata. Selleks on laboris kasutusel Cisco kommutaator SFE1000P 8-port 10/100 Ethernet Switch: PoE/ fanless [47]. Labori topoloogia on toodud joonisel 10.



Joonis 10. Andmeside kursuse labori topoloogia joonis

4.1 Laboritöö teoreetiline osa

Labori see osa on õppejõu poolne ettekanne tudengitele, kus antakse ülevaade SNMP toimimise mehhanismidest, võrguhaldusjaamadest (ja lähemalt Nagiose kasutamisest) ning seletatakse labori topoloogiat ja üliõpilaste poolt tegema hakatavaid ülesandeid.

Kõige suurema osa ettekandest moodustab SNMP tutvustamine üliõpilastele ja selle juures mainitakse ära ka, mis käskudega (õigem oleks öelda programmidega, kuid üliõpilaste jaoks parema arusaamise eesmärgil kasutatakse pigem sõna käsk) saab kindlaid SNMP protsesse läbi viia Linuxis kuna just Linuxi distributsiooni peal töötab laboris kasutatav võrguhaldusjaam.

Üldiselt tähendab labori tegemine üliõpilase jaoks mingi materjali iseseisvat läbi töötamist ja seejärel poolteist tundi praktiliste harjutuste tegemist. Materjali eelnev läbi töötamine on äärmiselt vajalik, sest see annab tudengile laboris tehtava kohta baasteadmised, et selle vähese pooleteise tunni jooksul, mis labori tegemiseks antud, saaks tegeleda ülesannetega, mitte ei raisata aega laboris uuritava asja idee arusaamiseks. Äärmiselt vajalik oleks kasvõi see, et tudeng omaks ülevaadet või põhimõttelist arusaama laboris uuritava kohta (mõnikord räägitakse laboris uuritava asja kohta ka loengus, kuid üldiselt siiski pigem väga vähe ja juhtub ka, et loengutesse paljud tudengid üldse ei jõuagi). Paraku näitab reaalne elu seda, et kui mingi eelnev materjal on vaja läbi töötada, siis üliõpilased tavaliselt seda ei tee- magistritöö autor teab seda rääkida oma kogemuste kui ka kaastudengite näite põhjal. Põhjus pole alati selles, et üliõpilased ei viitsiks materjali läbi töötada vaid täiesti reaalne ongi, et neil ei ole selleks aega- töö, teiste õppeainete ja ka pere kõrvalt on see ka igati mõistetav-, lihtsalt unustatakse see ära või ei teatagi, et midagi peaks eelnevalt iseseisvalt läbi töötama. Õnneks leidub siiski mõningaid tudengeid, kes on vajaliku materjali enne laborisse tulekut läbi vaadanud. Seega on laboris üliõpilased, kes laboris uuritavast midagi teavad ja saavad asjade käigust aru ning üliõpilased, kes ei tea asjadest mitte midagi. Taoline olukord viib aga selleni, et õppejõud peab hakkama laboris uuritavat täiesti algusest seletama neile, kes pole materjali läbi töötanud ning tõenäoliselt abistama just neid üliõpilasi. Samas kannatavad selle all üliõpilased, kes on materjali läbi töötanud ning kui neil tekib küsimusi, siis ei jõua õppejõud neid aidata. Loomulikult ei pruugi tudengid, kes pole materjali läbi töötanud, üldse midagi küsida, sest nad näevad, et teised oskavad või vähemalt teavad oluliselt rohkem ja ei tahagi midagi küsida. Selline olukord viib jällegi probleemini, et mõni tudeng saab asjadest aru ja teeb oma ülesanded ilusasti ära, kuid teine istub lihtsalt poolteist tundi laboris ja ei oska lõpuks uuritavast asjast midagi ja kõige halvemal juhul ei

saagi aru, millest üldse jutt käib. Eelnevalt toodud olukordade vältimiseks leiabki magistritöö autor, et üliõpilaste taset peaks enne labori praktilise osa tegemist ühtlustama, et kõikidel tudengitel oleks enne laboris arvuti taha istumist SNMP-st põhimõtteline arusaam ning nad saavad üldiselt aru, mida harjutuses küsitakse. Taseme ühtlustamiseks olekski ettekanne labori ajast, kus tudengid nagunii peaksid kõik kohal olema. Loomulikult võivad mõnel tudengil olla eelnevalt juba head teadmised SNMP-st või ta on enne laborisse tulekut ise vabatahtlikult asja uurinud, kuid see on tema jaoks ainult positiivne kuna ta saab ülesanded kiiremini tehtud või SNMP-d lihtsalt põhjalikumalt ise uurida.

Ettekande eesmärk SNMP ja võrguhaldusjaamade seisukohast pole terve magistritöö peatükkide „1. Võrguhalduse protokoll SNMP“ ja „2. Monitooringuvahendid“ üliõpilastele ette lugemine või detailne selgitamine vaid nimetatud peatükkidest kõige olulisematest asjadest ülevaate andmine. Väga positiivne oleks muidugi kõikide asjade pikemalt ja põhjalikumalt lahti seletamine, kuid kuna aega on umbes 20 minutit, siis tuleb teha piisavalt lühidalt, sest üliõpilastele peab jääma aega seda kõike ka praktikas läbi proovida.

Ettekande SNMP osa lahtiselgitamine algabki antud protokolli üldsõnalise defineerimisega. Edasi liigutakse SNMP üksuste kirjeldamisega ja üksuste vahelise suhtluse näitamisega. Siinkohal on ka oluline selgitada, mis porte SNMP üksuste vahelisel suhtlemisel rakendatakse ning mis protokolli ja miks just seda protokolli kasutatakse. Seejärel tutvustatakse haldusinfobaasi ja haldusinfostruktuuri ning selgitatakse, mis on nende eesmärk ja kuidas need on üles ehitatud. Pärast seda räägitakse SNMP protsessidest, mis ongi vahest labori praktilise osa seisukohast kõige olulisem. Siinkohal näidatakse ka mis käsku tuleb võrguhaldusjaamas kasutada, et kindel protsess rakenduks. Samuti seletatakse, mis võtmeid käsu juures üldiselt kasutatakse ja mis need tähendavad. SNMP osa lõpetuseks antakse ülevaade protokolli erinevatest versioonidest ning kindlasti räägitakse ka SNMP turvalisusest. Kuna laboris tuleb tegemist ka Nagiosega, siis antakse kiire ülevaade ka sellest võrguhaldusjaama tarkvarast ja näidatakse, mille abil Nagios teostab päringuid ning kuidas peab jälgitava võrguseadme

või tööjaama Nagioses määratlema. Lisaks mainitakse ka teisi magistritöös tutvustatud võrguhaldusjaamu, et tudeng teaks, missuguseid populaarseid NMS-e veel maailmas kasutatakse. Lõpuks näidatakse tudengitele ka labori topoloogiat ja selgitatakse lühidalt, mis ülesanded neid laboris ees ootavad. Selline ettekanne peaks autori arvates andma igale tudengile piisava arusaama SNMP alustest ja laboris toimuvast. Pärast ettekande kuulamist saavadki üliõpilased värskelt omandatud teadmisi praktikas rakendada alles jäänud aja jooksul.

4.2 Laboritöö praktiline osa

Selle osa jooksul sooritab üliõpilane iseseisvalt, vajadusel ka õppejõu juhendamisel, harjutusi. Praktilise osa jaoks antakse üliõpilasele ette laboritöö juhend, mis sisaldab informatsiooni labori ja laboris tehtava kohta. Labori juhendi struktuur on koostatud Andmeside kursusel hetkel kasutatava võrguhalduse labori põhjal, mis on kätte saadav allikast [40].

Laboritöö juhendis on esmalt kirjas andmed labori kohta. Nendeks andmeteks on labori pealkiri, laboriga seotud õppeaine, laborit juhendavad õppejõud ja laboritöö eesmärk koos harjutustes kasutatavate seadmete ja programmidega . Labori pealkiri on: Võrguhaldus *Simple Network Management Protocol* baasil. Õppeaine, millega magistritöös koostatav labor seotud on, kannab nime Andmeside ja selle kood on IRT0030. Labori on koostanud ning seda juhendab Märt Erik. Laboritöö eesmärgiks on tutvuda võrguhalduse jaoks kasutatava protokolliga SNMP ja seadistada ning tööle rakendada võrguhaldusjaama tarkvara Nagios. Laboris kasutatakse arvuteid operatsioonisüsteemiga Windows 7, millel töötab virtuaalserver, kus on kasutuses Linuxi operatsioonisüsteemi distributsioon Fedora *release* 12. Nagiose tarkvara töötab Fedora peal. Laboris on kasutusel ka Cisco kommutaator SFE1000P 8-port 10/100 Ethernet Switch: PoE/ fanless. Virtuaalserveri käivitamiseks kasutatakse programmi Vmware Player. Võrgust pakettide kogumiseks on programm Wireshark.

Pärast labori andmete kirjeldust on juhendis kirjas töö käik, mis koosneb ettevalmistusest labori alustamise jaoks, ülesannetest ja küsimustest, labori lõpetamise protseduuridest ja aruande ülevaatest. Töö käigust on pikemalt juttu antud paragrahvi järgnevates punktides ja alapunktides.

Laboritöö juhendi viimased osad on lühike õpetus Wiresharki kasutamise jaoks ja kasulikud lingid laboriga seotud asjade kohta. Õpetust Wiresharki kasutamiseks läheb vaja, sest labori jooksul peavad üliõpilased harjutuste käigus pakette püüdma ja neid analüüsima. Wiresharki õpetus on võetud hetkel kasutatava Andmeside kursuse võrguhalduse laborist leheküljelt [40] kuna sealne juhend on autori arvates asjalik ja annab kiire ülevaate programmi kasutamisest. Kasulike linkide all on paar internetiviidet SNMP kohta, kust saab juba põhjalikuma ülevaate protokollist. Samuti on sinna pandud Nagiose ja Nagiose dokumentatsiooni koduleht ning labori teoreetilise osa slaidide asukoht.

4.2.1 Töö käik

Laboritöö juhendis olev töö käik osa ongi konkreetselt see, mis üliõpilasel tuleb labori jooksul läbi teha ja õppejõule kaitsmiseks esitada. Aruanne on samuti labori töö üks osa ja kes on kiire saab selle harjutusi tehes valmis, ära vormistada veebilehena ning ette näidata. Tavaliselt on labori jooksul siiski piisavalt palju teha ja uurida, nii et aruanne kirjutatakse valmis üldiselt iseseisvalt pärast laborit, et esitatud küsimused saaksid korrektselt vastatud ja harjutused põhjalikumalt läbi analüüsitud. Analoogiliselt teiste aines tehtavate laboritega peab aruanne vastama World Wide Web Consortiumi (W3C) poolt esitatud standarditele.

Järgnevalt on kirjas magistritöö autori nägemus Andmeside kursuse jaoks mõeldud võrguhalduse labori ülesannetest. Ülesanded on koostatud selliselt, et üliõpilane peaks ise uurima SNMP komponente ja viima läbi SNMP protsesse ning uurima selle käigus, mis võrgus olevatel seadmetel toimus. Samuti peab tudeng vaatlema võrgus liikuvaid SNMP

pakette, et aru saada, mis kujul ja kuidas protokoll andmeid edastab. Oluline on ülesannete kaudu näidata ja rõhku pöörata ka SNMP turvalisusele, sest võrgus liikuvad andmed on tihti konfidentsiaalne informatsioon ja vähene turvalisus võib põhjustada ettevõttele suurt kahju. Lähemalt tutvustatakse ka ühte võrguhaldusjaama tarkvara, et tudeng teaks kuidas võrguseadmeid lihtsalt jälgida. Laboris praktilise töö käigus omandatud teadmiste tulemusel peaks iga üliõpilane olema võimeline selgitama SNMP tööpõhimõtet ning komponente ja vajadusel korraldama võrguseadmete jälgimist kas siis enda tarbeks või töökohas. Magistritöö autor koostab ülesanded tuginedes enda kogemustele ja maailma ülikoolide näidetele. Lisaks põhjendab käesoleva töö autor iga ülesande põhimõtet ning vajalikkust tudengi jaoks.

4.2.1.1 Ettevalmistus

Tudeng peab minema oma töökoha arvutis C kettal olevasse kausta SNMP_labor. Sealt seest tuleb kopeerida kaust Nagios_32bit arvuti töölauale (*desktop*ile). Samal ajal kui toimub kopeerimine tuleb tudengil teha kindlaks arvuti IP aadress, sest seda läheb edaspidi laboris vaja. IP aadressi saab kätte vajutades töölaual *Start* ning trükkides käsu *cmd*. Avanenud aknas peab trükkima *ipconfig* ning otsima üles IPv4 aadressi. Labori arvutitele jagatakse IP aadressid automaatselt dünaamilise hostikonfiguratsiooni protokolliga (DHCP) vahemikust 192.168.252.160 kuni 192.168.252.200. Lisaks peaks kontrollima, kas laboris kasutatav Cisco kommutaator aadressiga 192.168.0.60 on arvutist kätte saadav- selleks peab trükkima käsu *ping* 192.168.252.60 ning vaatama, kas kommutaator vastab päringule.

Tudeng peab veenduma, et arvutis töötab SNMP teenus ja see on õppejõu poolt määratud nõuete järgi seadistatud, sest ülesannete käigus hallatakse SNMP-ga ka labori arvuteid. Teenust saab näha vajutades *Start* ja trükkides *services.msc* või tehes *Computer* peal hiirega parem klõps ja valides *manage* ning avanenud aknas *Services and Applications* alt valida *Services*. Teenuste aknas tuleb üles otsida SNMP *Service* ja vaadata kas *Status* all on *Started*. Juhul kui teenus on välja lülitatud, siis peab teenuse peal tegema hiirega

parema klõpsu ja valima *Start*. Seadistuste kontrollimiseks tuleb SNMP teenuse peal jällegi teha hiirega parem klõps ja valida *Properties*. Saki (*tab*) *Agent* all *Service* sektsioonis peab olema valitud *Physical* ja *Contact* ning *Location* juurde peab üliõpilane vabal valikul mingi kontakti ja asukoha kirjelduse panema. Saki *Security* all *Accepted community names* sektsioonis tuleb määrata kogukonnanimi *public* õigustega *READ WRITE*. Lisaks tuleb valida *Accept SNMP packets from these hosts* ja lisada sinna oma Nagiose virtuaalserveri aadress, et päringuid saaks teostada ainult kindla IP-ga võrguhaldusjaam. Nagiose virtuaalserveri IP aadress antakse igale tudengile praktilise osa alguses. Üliõpilane viib läbi haldustegevusi ka Cisco kommutaatoriga, kuid selle on labori juhendaja juba eelnevalt ise ära seadistanud.

Pärast Nagiose virtuaalserveri kopeerimist arvuti töölauale ja SNMP teenuse seadistamist Windowsi tööjaamas peab tudeng virtuaalmasina käivitama. Selleks tuleb esmalt avada programmide alt VMware Player ja valida *Open a Virtual Machine*. Seejärel peab tudeng avanenud aknas üles otsima töölauale kopeeritud kaustast *.vmx* laiendiga faili ning vajutama *Open*. Seejärel valima VMware Playeris virtuaalmasina Nagios_32bit ja vajutama *Play virtual machine*. Virtuaalmasin käivitub minuti jooksul ning pärast seda peab tudeng virtuaalserverisse sisse logima. Kasutaja on nagios ja parool on samuti nagios. Magistritöö autor soovitab siiski virtuaalserverisse sisse logida mõne SSH kliendiga, milleks võiks olla näiteks Putty. Põhjus on asjaolus, et SSH kliendi abil on palju mugavam virtuaalserveris ülesandeid läbi viia kui VMware Playeri konsooli kaudu. Pärast sisselogimist on ettevalmistused labori jaoks tehtud ja üliõpilane saab hakata sooritama SNMP ja Nagiose ülesandeid.

4.2.1.2 SNMP-ga informatsiooni pärimine

Kuna SNMP päringud on kõige elementaarsemad ja enam kasutatavad toimingud võrguhalduse protokolli juures, siis tulekski alustada SNMP tutvustamisega just päringute kaudu. Kõige tavalisem SNMP päring viiakse läbi *get* protsessiga- mida Linuxis rakendab käsk *snmpget*- ning selle abil küsitakse üldiselt informatsiooni ühe kindla OID

kohta. Üliõpilaste esimeseks ülesandkes ongi pärida süsteemi kirjeldus: *snmpget –v 2c –c public arvuti_IP_aadress sysDescr.0*, kus võti –v määrab SNMP versiooni (kasutatakse alati viimast võimalikku versiooni) ja –c kogukonnanime. Vastuseks saadaksegi päritud masina kirjeldus näiteks kujul: *SNMPv2-MIB::sysDescr.0 = STRING: Hardware: x86 Family 6 Model 15 Stepping 11 AT/AT COMPATIBLE - Software: Windows Version 6.1 (Build 7600 Multiprocessor Free),* kus on kirjas ka OID-d sisaldav MIB. Päringu eesmärgiks ongi näidata, kuidas lihtne SNMP päring ja vastus toimib. Tudengile esitatakse ka küsimus, mis MIB-is asub objekt *sysDescr* ja lastakse üles märkida ka päringu vastus. Kuna mõnikord on vaja teada lisaks tekstilisele kujule ka objekti numbrilist kuju, siis peab tudeng ka saadud tulemuse numbrilise kuju kätte saama kasutades *snmptranslate* käsku. Numbrilise OID teadasaamiseks on käsk kujul: *snmptranslate -On MIB_fail::objekti_nimi*.

Tihti tuleb ka ette olukordi, kus on vaja teada korraga MIB-i alampuu väärtusi või otsida terve tabeli informatsiooni. *Get* protsessiga oleks seda väga tülikas teha. Suure koguse info pärimiseks on käsk *snmpwalk*, mis rakendab *getnext* protsessi. *Snmpwalk* käsu põhimõte on selles, et päringuks tuleb ette määrata alampuu või huvi pakkuvate objektide kasvõi osaline objekti nimetus. Antud juhul lastakse üliõpilasel pärida *ifPhysAddress* objekte kommutaatorist, mis annavad vastuse seadme võrguliideste füüsiliste aadressite kohta. Samas päringul esitatakse ainult osaline objekti nimetus näiteks kujul *ifPh*. Päring esitatakse seega kujul *snmpwalk –v 2c –c public 192.168.252.60 ifPh* ning vastused saadakse kujul *IF-MIB::ifPhysAddress.1 = STRING: 9c:4e:20:1f:6c:52,* mis sisaldab jällegi MIB-i, kus objekt asub ja võrguliidese füüsilist aadressi. Neid vastuseid on nii palju kui on kommutaatoril võrguliideseid. Erinevate MIB-ide tundmaõppimiseks küsitakse jällegi üliõpilaselt, mis MIB-is asuvad päritavad OID-d ning mitu võrguliidest on seadmel. Viimase küsimuse vastuseks on erinevate füüsiliste võrguliideste aadresside arv.

Nagu kirjutatud, siis *snmpwalk* käsk rakendab *getnext* protsessi, mis on suurte infohulkade pärimisel ebaratsionaalne, sest iga objekti jaoks tehakse uus päring. Selles veendub tudeng ka ise, sest enne *snmpwalk* käsu tegemist palutakse tal võrguliiklust

kommutaatori suunas jälgida Wiresharkiga. Üliõpilaselt küsitaksegi mitu päringut ja vastust kokku oli füüsiliste aadresside küsimisel, mis protsessi kasutab *snmpwalk* käsk ja miks ei ole seda mõistlik kasutada suurte SNMP andmehulkade pärimisel ja kuidas saaks suurt hulka andmeid ratsionaalsemalt küsida võrguseadmelt. Vastuseks viimasele kahele küsimusele ongi asjaolu, et *snmpwalk* on väga ressursimahukas toiming võrguliikluse ja võrguhaldusjaama ning seadme koormuse poole pealt kuna iga objekti info saamiseks tehakse uus päring. Samas kasutades *snmpbulkwalk* käsku rakendab see *getbulk* protsessi, mille puhul pannakse vastustesse nii palju informatsiooni kui seadme SNMP agent võimaldab. Sellega hoitakse suurel hulgal kokku nii võrguliikluse mahtu kui ka seadme ressursse. Üliõpilane peab antud vastuse teadmiseks lugema natukene ise SNMP kohta ja selleks ongi materjalidele viide labori lõpus „Kasulikud lingid“ all.

Igal pool laboritöö käigus, kus üliõpilane kasutab võrguliikluse jälgimiseks Wiresharki, palutakse tudengil ka salvestada Wiresharki fail, mis on .pcap laiendiga. Selle mõte on asjaolus, et kui üliõpilane ei jõua labori ajal kõikidele aruandes soovitud küsimustele vastuseid kirja panna või soovib tudeng hiljem pikemalt uurida tehtud tööd, siis saab ta näiteks kodus avada salvestatud faili oma arvutisse paigaldatud Wiresharki programmiga ja näha täpselt seda, mis laboris sai tehtud.

4.2.1.3 SNMP-ga informatsiooni muutmine

Lisaks SNMP päringutele on võimalik võrguhalduse protokolliga ka teatud OID-de väärtusi muuta. Selleks on kasutusel protsess *set* ja antud protsessi rakendab käsk *snmpset*. Käsk on samasuguse kujuga nagu *snmpget*, kuid lisaks tuleb käsu lõppu lisada objekti väärtuse tüüp (ainult üks täht) ja määratav väärtus ise: *snmpset –v 2c –c public seadme_IP_aadress OID OID_tüüp OID_uus_väärtus*. Objekti väärtuse tüüpideks on näiteks sõne, mida tähistatakse tähega s (inglisekeelsest sõnast *string*), täisarv, mida tähistatakse tähega i (inglisekeelsest sõnast *integer*) ja IP aadress, mida tähistatakse tähega a. Muuta saab siiski ainult nende objektide väärtusi, kuhu MIB failis on OID juures *MAX-ACCESS* kohta kirjas *read-write* või *read-create*.

Üliõpilase ülesandeks *set* protsessi juures on muuta arvutis olevad süsteemi kontakti ja asukoha kirjeldus. Need on kirjas arvutis SNMP teenuse atribuutides *Agent* saki all ehk väärtused, mis tudeng ise labori ettevalmistuse alapunkti „4.2.1.1 Ettevalmistus“ all arvutile määras. Tudeng peab ise välja uurima SNMPv2-MIB failist, mis on asukoha ja kontakti kirjelduse OID, ning mis on nende objektide väärtuse tüüp. Tüübi leidmiseks peab tudeng kasutama *snmpset* juhendit, milleks on käsk *man snmpset*. Juhendist väljumiseks tuleb lihtsalt trükkida täht q. Asukoha väärtuseks tuleb siinkohal panna arvuti_laboris ja kontakti väärtuseks tudeng@SNMPlaboris. Üliõpilasele antakse ette *snmpset* käsu kuju, seega lõpliku käsu peab ta ise valmis kirjutama. Päring ja vastus tuleb panna kirja aruande jaoks. Loomulikult peab üliõpilane ise veenduma ka, et süsteemi asukoha ja kontakti kirjeldus on muutunud arvutis.

Kontakti kirjelduse seadmise kohta oleks õige käsk *snmpset -v 2c -c public arvuti_IP_aadress sysContact.0 s tudeng@SNMPlaboris* ja sellele saadetav kinnitus *SNMPv2-MIB::sysContact.0 = STRING: tudeng@SNMPlaboris*. Nende objektide muutmine ei mõjuta arvuti töös mitte midagi ja on lihtsalt üliõpilasele ülevaate andmiseks kuidas SNMP abil teatud seadistustega saab manipuleerida. Samas on võimalik muuta ka objekte, mis mõjutavad seadme tööd oluliselt. Näiteks saab *snmpset* käsuga katkestada kommutaatori/ marsruuteri võrguliidese töö ja sellega võtta ühendus ära kindlatel arvutitel või isegi tervel võrgusegmendil. Sellist situatsiooni tudeng hilisemas ülesandes ka katsetab.

4.2.1.4 SNMP turvalisus

SNMP versiooni 1 ja 2 ainukeseks turvalisuse elemendiks on kogukonnanimi, mida kasutatakse kui paroolina autentimisel. Samas toimivad SNMPv1 ja SNMPv2 puhul kõik päringud ja vastused avatud tekstina ehk kui keegi peaks liiklust pealt kuulama, siis saadakse teada nii kogukonnanimi kui ka kogu võrgus liikuv SNMP informatsioon. See võib tähendada aga olulise võrguhalduse informatsiooni sattumist valedesse kätesse või

isegi interneti käideldavuse kadu- kui kurjategija teab kogukonnanime ja marsruuteris on lubatud SNMPv1 või SNMPv2 puhul *set* protsess, siis võib ta välja lülitada internetti tagava võrguliidese. Samas SNMP versioon 3 tagab võrguhalduse informatsiooni turvalisuse (õigemini tagab SNMPv3 informatsiooni turvalisuse kaks komponenti- tervikluse ja konfidentsiaalsuse-, kuid mitte käideldavust ja seega ei anna SNMP versioon 3 täielikku turvalisust. Lihtsuse mõttes on siiski antud magistritöös öeldud, et SNMPv3 tagab turvalisuse, sest võrgust ei ole võimalik andmete autentimiseks ja krüpteerimiseks vajalikke paroole kätte saada ning SNMP haldusinformatsioon liigub krüpteeritud kujul).

Tudengitele ongi äärmiselt oluline demonstreerida turvamata ja turvalise SNMP versiooni erinevusi ning panna üliõpilased mõtlema, et kui asju saab teha turvaliselt, siis neid võimalusi peaks ka rakendama. Kahjuks saab enamik inimesi sellest aru alles siis, kui esimene turvalisuse kadu on juba toimunud. Loomulikult nõuab SNMPv3 funktsioonide rakendamine ka natuke rohkem seadistamist (ja teadmisi), kuid see ajakulu on autori arvates piisavalt väike võrreldes võimalike kahjudega.

SNMP versiooni 2 ja 3 erinevuste väljatoomiseks lastakse tudengil pärida turvamata ja turvalise versiooniga kommutaatorist lihtsalt süsteemi nime OID-ga *sysName.0*. See ei ole küll turvalisust vajav informatsioon, kuid mõte on siinkohal näidata üliõpilasele päringute erinevusi ning samuti lasta tudengil uurida, mida tähendavad versiooni 3 juures päringute tegemisel lisandunud võtmed/elemendid ning mis räsi- ja krüpteerimisalgoritme saab SNMPv3 juures kasutada ja mis algoritme kasutatakse vaikimisi. Päringud tehakse just kommutaatori suunas kuna Windowsi operatsioonisüsteemides töötavad SNMP agendid ei toeta turvalist SNMP versiooni 3 [58]. Kommutaatoris on eelnevalt turvalisuse funktsioone tagavad seadistused labori juhendaja poolt tehtud.

SNMPv2 päringul kommutaatori süsteemi nime küsimiseks on käsk *snmpget -v 2c -c public 192.168.252.60 sysName.0* ning vastus sellele tuleb järgnev: *SNMPv2-MIB::sysName.0 = STRING: SNMP_labori_switch*. Samas näeb igaüks, kes võrku jälgib, et paketis on näha kogukonnanimi, milleks on *public* ja andmeid OID kohta

ehk *SNMPv2-MIB::sysName: SNMP_labori_switch*. Kogukonnanime alusel saab juba kurjategija soovitud informatsiooni ise pärida või seada. SNMPv3 päring kommutaatori nime küsimiseks on *snmpget -v 3 -l authPriv -u snmpv3 -A esimeneparool -X esimeneparool 192.168.252.60 sysName.0* ja vastus pärijale on samasugune nagu SNMPv2 puhul. Siinkohal aga ei avaldata võrguliikluses kuskil autentimise ja krüpteerimise jaoks vajalikke paroole ning SNMP andmete osa ehk PDU on näiteks krüpteeritud kujul: *encryptedPDU: 7CC189FC579112E96BF442BEB1E460BBFF*.

4.2.1.5 SNMP *trap*

Trap protsessi puhul on tegemist süsteemiga, kus ei toimu võrguhaldusjaama poolt võrguseadmete poole päringuid vaid seade saadab ise NMS-ile teate kui mingi sündmus on masinas toimunud. Selliseks sündmuseks võib olla näiteks kindla võrguliidese töö katkemine või positiivsel juhul just liidese tööle rakendumine.

Magistritöö laboris saadab Cisco kommutaator üliõpilase võrguhaldusjaamale *trap*'i, mille nimeks *authenticationFailure*. See *trap* saadetakse juhul kui seadmest päritakse SNMP informatsiooni vale kogukonnanimega. *Trap*'i tekitajaks on tudeng ise, sest kõigepealt ongi tema ülesanne pärida kommutaatorist suvalist objekti SNMPv2-ga, kuid päringu puhul ei tohi olla kogukonnanimi *public*. Kommutaatori SNMP agent näeb, et informatsiooni üritati pärida vale kogukonnanimega ja saadab kohe seadmes määratud võrguhaldusjaamadele *trap* teate. Reaalsuses võib sellisest teatest palju kasu olla kuna võrguhaldusega tegelev inimene näeb, et keegi üritab infrastruktuuri seadmete kohta informatsiooni küsida. Päringute teostamisel ja *trap* teate saamisel peab üliõpilasel töötama Wireshark, et püüda kinni just *trap* pakett. Tudeng peab salvestama ekraanipildi, kus on näha *trap*'i andmed OID kohta numbrilisel kujul kui ka sõnalisel kujul ning selgitama lahti antud *trap*'i tähenduse.

4.2.1.6 Ülesanded

Ülesannete osa mõte laboritöös on näidata tudengitele reaalset SNMP kasu/ vajalikkust võrguhalduse toimingute läbiviimisel. Ülesannete alapunkti alla laboris ei kirjutata enam tudengile konkreetselt ette millist käsku kasutada või mida pärida (siiski antakse ette mõni vihje, et oleks vähemalt alguspunkt, kust lahendust otsima hakata). Siinkohal kirjeldatakse üliõpilasele ette kindel olukord, mis võrguhalduse administraatoril võib tekkida ja tudeng peab antud situatsiooni lahendama või situatsiooni lahendamiseks piisavalt palju informatsiooni koguma kasutades selleks ainult protokolli SNMP. Üliõpilane peab ise analüüsima situatsiooni ja mõtlema, mis informatsiooni tal on vaja olukorra lahendamiseks- selleks peab ta uurima MIB faile, et teada millise või milliste objektidega on situatsioon seotud ning kasutama õigeid SNMP protsesse rakendavaid käske. Igat olukorda saab arvatavasti lahendada ka teiste vahenditega peale SNMP, kuid samas võib võrguhalduse protokolliga kõige kiiremini ja mugavamalt vajaliku informatsiooni kätte saada või olukorra lahendada.

Esimene ülesanne on tehtud Boston University labori neljanda osa esimese harjutuse põhjal, kus peab uurima kindla seadme suunas loodud TCP ühenduste andmeid (täpsemalt öeldes tuleb välja selgitada Telnet ühenduse parameetrid ja oluline on teada saada algataja IP aadress). Käesoleva magistritöö esimese situatsiooni puhul soovib võrguadministraator muuta kommutaatoris seadistusi, kuid SSH-ga sisse logida üritades (kõik administraatorid kasutavad seadme haldamiseks SSH-d) antakse talle teade, et korraga saab seadet hallata ainult üks kasutaja (sest selline on näiteks kindla kommutaatori eripära või firma turvapoliitika nõuab seda, kuid see ei oma antud ülesande juures tähtsust). Kuna ettevõttes on mitu inimest, kellel on õigus kommutaatorit administreerida, siis järelikult on keegi juba seadmesse sisse loginud (tehes seal siis muudatusi või on lihtsalt ühenduse püsti unustanud). Samas on administraatorile väga oluline muuta kiiremas korras kommutaatoris kindlad seadistused. Kuna sisselogimist keelavas teates ei ole öeldud, mis IP aadressi pealt on kommutaatori suunas SSH ühendus või ühendused loodud (ühenduse võivad olla loonud mitu inimest, kes tahavad kõik seadet hallata, kuid ainult üks nendest on reaalselt sisse loginud), siis peab administraator

selle välja uurima ning ta teeb seda kasutades SNMP-d. Pärast IP aadressi või aadresside teadasaamist helistab näiteks administraator vastava arvuti taga töötavale inimesele ja palub ühendus lõpetada, kuid see osa pole enam oluline antud ülesande seisukohast.

Tähtis on teada saada aktiivsete ühenduste IP aadress või aadressid. Seda saab teha kasutades *snmpwalk* käsku ja uurides haldusinfobaasis TCP alampuud kuna SSH kasutab transpordikihil just TCP-d: *snmpwalk -v 2c -c public 192.168.252.60 tcp* (loomulikult võib üliõpilane kasutada ka SNMP versiooni 3 päringuid). Saadud tulemuste juures tuleb ostida objekti *tcpConnLocalPort*, mille väärtus on 22, sest SSH ühendus tehakse porti 22. Seejärel peab otsima nimetatud objektile ja pordile vastavat objekti *tcpConnRemAddress* ning selle väärtuseks ongi ühenduse looja arvuti IP aadress. Kuna SSH ühenduse kommutaatori suunas loovad üliõpilased ise, siis võib olla ka mitu objekti, mille väärtus on 22 ja arvutite IP aadressid. Tudeng peab need kõik ekraanipildi peal näitama ja ka ise kirja panema. Lisaks peab üliõpilane kirja panema ka terve *snmpwalk* käsu aruande jaoks. Vihjena antakse TCP alampuust informatsiooni otsimine, mis peaks tudengi olukorra lahendamiseks õigele teele viima.

Teine ülesanne on koostatud näitamaks üliõpilasele, et *set* protsessi saab ka tõsisemateks operatsioonideks kasutada kui ainult süsteemiinfo ja asukoha kirjelduse muutmine, mida tehti alapunktis „4.2.1.3 SNMP-ga informatsiooni muutmine“. Teise situatsiooni puhul näeb administraator tulemüüri logikirjetes, et üks kindel arvuti ettevõttes on viirusega nakatunud ja saadab pidevalt erinevatele avalikele IP aadressidele mingisugust informatsiooni. See arvuti tuleb võimalikult kiiresti võrgust isoleerida. Administraator teab, missuguse võrguliidese küljes see kindel arvuti on ja seetõttu otsustab ta isoleerimiseks võrguliidese välja lülitada. Situatsiooni lahenduseks on *snmpset* käsu kasutamine ja liidese, mille küljes on arvuti, administratiivse staatuse välja lülitamine (võrguliidesel on kaks töötamist iseloomustavat staatust/ olekut - administratiivne ja operatsiooniline ehk vastavalt inglise keeles *administrative* ja *operational*).

Operatsiooniline staatus tähendab seda, kas liides edastab liiklus või mitte, vastavad olekud siis *up* või *down*, administratiivne staatus tähendab seda, kas liides on sisse

lülitatud või mitte, samuti vastavad staatused *up* või *down*. Liidese administratiivne staatus ei sõltu operatsioonilisest staatusest, kuid operatsiooniline staatus sõltub administratiivsest staatusest. Juhul kui administratiivne staatus on *down*, mida ka antud ülesande lahendusena taotletakse, siis on igal juhul ka operatsiooniline staatus *down* ehk liides ei saa enam liiklust edastada.

Olukorra lahendamiseks tuleks tudengil rakendada käsk: *snmpset -v 2c -c public 192.168.252.60 ifAdminStatus.liidese_number i 2*, i tähistab objekti *ifAdminStatus* väärtuse tüüpi ehk täisarvu ja 2 tähendab liidese välja lülitamist ehk staatust *down*. Tudeng saab tulemust kontrollida kasutades oma arvutist *ping* käsku viirusega arvuti suunas: *ping –t viirusega_arvuti_IP_aadress*. Pärast liidese välja lülitamist peab tudeng liidese uuesti sisse lülitama, et teised üliõpilased saaksid antud situatsiooni lahendada ja aruande jaoks ka *snmpset* käsud kirja panema. Võib juhtuda, et kõik tudengid satuvad antud ülesannet samal hetkel tegema, mis tähendab seda, et üks inimene tegeleb asjaga ja ülejäänud viis põhimõtteliselt ootavad. Selle asemel on kasutusel kaks liidest ja kaks viirusega arvutit, et maksimaalselt kolm üliõpilast peaksid ühte liidest välja lülitama ning ainult kaks peaks tema järgi ootama. Tudengid, kelle virtuaalserverite IP aadressid on 192.168.252.61, 192.168.252.62 ja 192.168.252.63 *ping*ivad viirusega arvutit, mille IP on 192.168.252.70 ning mis asub kommutaatori seitsmenda võrguliidese küljes. Ülejäänud tudengid *ping*ivad viirusega arvutit, mille IP aadress on 192.168.252.71 ja mis asub kommutaatori kaheksanda võrguliidese küljes. Vihjena öeldakse tudengile, et vajaliku informatsiooni leiab IF-MIB failist ja tuleb otsida liidese staatust haldavat objekti.

Kolmanda ülesande eesmärk on lasta uurida tudengil liikluse mahtu erinevate võrguliideste peal ning teha väike arvutusülesanne leidmaks kõige koormatuma liidese liikluse hulka megabittides. Arvutusülesande eesmärk on vaadata, kas üliõpilane saab õigesti aru päritavast informatsioonist kuna liikluse maht esitatakse oktettide arvuna ehk baitides (1 bait on võrdne 8 bitiga ehk ühe oktetiga), kuid tulemust soovitakse bittides. Kolmanda situatsiooni puhul on administraatori soov teada saada kommutaatori võrguliideseid läbinud liikluse hulka, et näha kui suur koormus seadmel erinevatel

liidestel on (seda informatsiooni võib vaja minna näiteks tasuvusarvutuste juures või liigse koormusega võrgusegmentide määramisel).

Tudeng peaks ülesande lahendamiseks kasutama *snmpwalk* käsku ja uurima haldusinfobaasi *interface* alampuud. Kuna nimetatud on alampuu annab väga palju informatsiooni, siis on asjalikum uurida *ifInOctets* ja *ifOutOctets* objekte, millega saab teada siis vastavalt liidese poolt vastu võetud baitide hulga ja liidesest välja saadetud baitide hulga. Snmpwalk käsud selle informatsiooni leidmiseks on: *snmpwalk -v 2c -c public 192.168.252.60 ifInOctets* ja *snmpwalk -v 2c -c public 192.168.252.60 ifOutOctets*. Üliõpilasel palutakse kirja panna käsud ja ekraanipildina näidata ka saadud tulemused. Lisaks küsitakse mis liidese pealt on kõige rohkem liiklust saadetud ja mis liidese pealt vastu võetud ning anda liikluse hulk nendel liidestel väljendatuna megabittides. Vihjena öeldakse siinkohal, et vajalikud objektid leiab IF-MIB-ist.

4.2.1.7 Nagiose seadistamine

Tudeng on siiamaani laboris vaadelnud SNMP toimimist ja uurinud erinevaid objekte MIB failides ning tutvunud ka SNMP versiooniga, mis tagab võrguhalduse käigus edastatava informatsiooni turvalisuse. Seda kõike on üliõpilane teinud käsitsi, mis reaalses võrguhalduses oleks mõeldamatu kuna seadmeid, mida jälgida, on infrastruktuuris arvatavasti päris palju ja kindlasti on soov iga võrguseadme peal teostada seiret rohkem kui ühele objektile. Siinkohal tulebki appi Nagiose tarkvara, mis teostab infrastruktuuri monitooringut automaatselt täpselt nendele seadmetele ja objektidele, mis on ette määratud. Jälgitava seadme või objekti probleemi korral teavitatakse võrguhalduse spetsialisti e-posti või sõnumi teel.

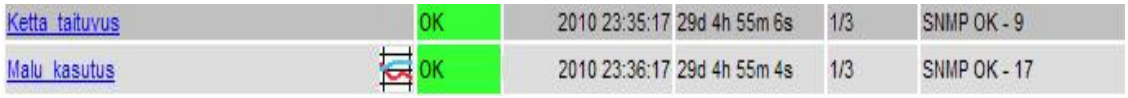
Labori viimase ülesandena peabki üliõpilane defineerima kaks jälgitavat masinat- üks tema enda labori arvuti ja teine Cisco kommutaator. Mõlema masina juurde tuleb kohe määrata ka *trap*'ide vastuvõtmiseks vajalik seadistus. Arvutile peab üliõpilane määrama

vabal valikul kaks jälgitavat objekti MIB-ist *HOST-RESOURCES-MIB.txt*. Valitud asjad tuleb ka lahti selgitada ning põhjendada, miks just need objektid valiti. *HOST-RESOURCES-MIB.txt*-st objektide valik arvuti jaoks on kasulik seetõttu, et antud MIB ongi mõeldud pigem arvutite ja serverite ressursside jälgimiseks. Cisco kommutaatori seireks tuleb üliõpilasel samuti määrata kaks jälgitavat objekti omal vabal valikul (seekord pole MIB-i ette määratud). Valik tuleb lahti seletada ja põhjendada selle objekti jälgimise kasulikkust kommutaatori seisukohast. Kuna Cisco kommutaator toetab SNMP versiooni 3, siis peavad olema ka päringud tehtud turvalisel kujul (Nagioses turvaliste päringute teostamiseks peab tudeng ise uurima *check_snmp* pistikprogrammi, et teada, missuguseid võtmeid ja mis väärtusi kasutada rakendamaks autentimise ja krüpteerimise funktsioone. Pistikprogrammi juhendit saab näha virtuaalserveri konsoolist järgmise käsu abil: */usr/local/nagios/libexec/check_snmp --help*). Lisaks peab tudeng tegema päringu objektile *ifOperStatus.9*, mis kontrollib kommutaatori üheksanda võrguliidese töösolekut. Mõte on siinkohal selles, et pärast tudengite poolt päringute seadistamist tõmbab õppejõud sellest liidesest kaabli välja ning üliõpilasele saadetakse e-posti peale teade, et antud võrguliides ei tööta (iga objekti juurde määrab tudeng ka oma e-posti aadressi). Pärast seda pannakse kaabel tagasi ning tudengile saadetakse jällegi e-kiri kinnituseks, et võrguliides toimib korrektselt. Lisaks saadetakse üliõpilasele ka *trap* teated liidese mitte funktsioneerimisest ja uuesti tööle hakkamisest. Sellega simuleeritakse näiteks reaalse elu olukorda, kus üheksanda võrguliidese taga asuvad väga olulised serverid ning kõik kommutaatori küljes olevad arvutid ei saa selle liidese katki minemisel enam nendele serveritele ligi. Võib juhtub ka, et liides ei ole ise katki läinud vaid näiteks keegi on kogemata kaabli sellest võrguliidesest välja tõmmanud. Pärast intsidenti saab võrguadministraator teate probleemist ja tal on võimalus kohe asja uurima hakata selle asemel, et keegi teatab veerand või pool tundi hiljem juhtunust. Selleks ajaks on võib-olla tekkinud juba oluline rahaline kadu ja loomulikult heidaks antud seik ka halba varju administraatorile endale kuna tema peaks probleemidest võrgus esimesena teadma.

Pärast seadmete ja objektide defineerimist peab tudeng Nagiose teenuse taaskäivitama, et muudatused rakenduks ja seadmeid jälgima hakataks. Seire tulemuse nägemiseks peab

tudeng veebibrauserisse sisestama Nagiose monitooringulehe aadressi, milleks on https://Nagiose_virtuaalserveri_IP_aadress/nagios ja sisse logima. Kasutajanimi on „nagiosadmin“ ja parool on „nagios“. Vasakult menüüst *Services* valides saab vaadata monitooringu tulemusi.

Joonisel 11 on näha kaks Nagiose jälgitavaid objekte võrguseadmel. Kirjeldusest saab aru, et tegu on kõvaketta mahu ja mälu kasutuse monitooringuga. Lisaks on näidatud ka viimase päringu ja objektide järjest pärimise aeg. Paremal pool on näha, et seadme ressursitarve on normis kuna kõvakettast on kasutusel ainult 9% ja mälu tarbitakse päringu hetkel 17%. Samamoodi näevad tudengid enda jälgitavaid objekte Nagiose monitooringulehel.



Joonis 11. Nagiose päringute tulemus

4.2.1.8 Töö lõpetamine ja aruande koostamine

Pärast kõikide harjutuste tegemist on üliõpilasel veel vaja teha paar asja enne laborist lahkumist. Esimese asjana peab tudeng välja lülitama virtuaalmasina. Selleks tuleb halduskonsoolis (Putty aknas või VMware Playeris) rakendada käsk *poweroff*. Seejärel tuleb kustutada töölaual olev Nagios_32bit kaust. Teise asjana peab üliõpilane kinni panema arvutis SNMP teenuse, et keegi ei saaks seal labori välisel ajal mingeid haldustegevusi rakendada. Selleks peab SNMP *Service*'i peal tegema hiirega parema klõpsu ja valima *Stop*. Sellega on labor lõpetatud ja üliõpilane võib lahkuda.

Aruande koostamisel tuleb kirja panna kõik asjad vastavalt laboritöö juhendi punktidele- päringud, käsud, ekraanipildid, Nagiose seadistused ja situatsioonide lahendused- labori juhendi nendest kohtadest, kus on öeldud panna kirja, märkida üles, tee või salvesta ekraanipilt. Lisaks tuleb tudengitel loomulikult vastata kõikidele laborijuhendis esitatud küsimustele ning kui kuskil on nõutud selgitamist või põhjendamist, siis tuleb ka seda

teha. Aruandele ei pea tudeng lisama Wiresharki .pcap laiendiga faile, sest need on üliõpilase enda jaoks abistavaks materjaliks labori aruande koostamisel.

5. Magistritöö edasiarendus

Käesoleva magistritöö eesmärk on koostada Andmeside kursuse jaoks võrguhalduse labor, mis baseerub protokollil SNMP ja tutvustada lähemalt tudengitele ka ühte võrguseire tarkvara, mis töötab võrguhaldusjaama ülesannetes, jälgides infrastruktuuris olevate seadmete tööd. Eelnevas peatükis ongi toodud labori kirjeldus. Autori arvates on laboris tehtavad ülesanded koos ettekandega piisavad selleks, et üliõpilane omaks ülevaadet SNMP toimimise mehhanismidest ja komponentidest. Samuti saab üliõpilane laboris praktilise kogemuse Nagiose tarkvaraga, millega on võimalik SNMP abil jälgida võrgus olevate seadmete tööd.

SNMP on äärmiselt kasulik protokoll kuna annab infrastruktuuris toimuva kohta väga palju informatsiooni, kuid siiski on temalgi piirangud. Võrguhaldus on laiahaardeline mõiste ja lisaks SNMP-le hõlmab ka näiteks erinevate teenuste (avalikud teenused nagu e-posti teenus SMTP protokolliga, veebiteenus HTTP protokolliga jne ning privaatsed teenused nagu Windowsi operatsioonisüsteemi spetsiifilised teenused) jälgimist ning toimimise kontrolli. Käesolevas peatükis tutvustataksegi magistritöö edasiarendamise võimalusi, et saaks jälgida erinevaid teenuseid ja sellega laiendada ning parendada võrguhaldust. Lisaks kirjeldab magistritöö koostaja ka päringute põhjal raporteerimise võimalusi graafikute alusel, et võrguhalduse spetsialistil oleks SNMP abil infrastruktuurist visuaalsem ülevaade ning selle arvelt ka lihtsam teha analüüse näiteks seadmete koormuse kohta. Edasiarenduse võimaluste juures on aluseks Nagiose tarkvara. Kuna laboris kasutatav virtuaalmasin on täielikult vabavaraline- nii seal peal töötava operatsioonisüsteemi kui ka tarkvara poolest-, siis võib loomulikult igaüks arendada magistritöö tulemust edasi täpselt nii nagu ise soovib, et täiustada võrguhaldust ja omada infrastruktuuri eri tahkude põhjalik ülevaade. Järgnevalt näitabki autor mõnda võimalust mõnda võimalust, kuidas lisaks juba loodud laborile saab võrgus toimuvast veel parema ülevaate.

5.1 Avalike teenuste jälgimine

Nagios kasutab monitooringuks pistikprogramme. SNMP puhul on selleks pistikprogrammiks *check_snmp*, mille abil viiakse läbi protsessid võrgus olevatest seadmetest info saamiseks. Samas saab SNMP-ga üldiselt informatsiooni võrgus toimuva liikluse ja seadmete parameetrite kohta, kuid infrastruktuuris avalike teenuste- näiteks veebiteenus, e-posti teenus, ajateenus jne- kohta info pärimises võib SNMP võimalustest väheseks jääda. Siiski oleks vajalik võrguhalduse käigus omada ülevaadet ka avalike teenuste toimimisest (ja mitte ainult toimimisest vaid kasulik oleks teada kas teenus toimib nii nagu ette antud nõuded määratlevad) ning selleks ongi Nagioses väga palju erinevaid pistikprogramme. Lisaks võib kindel teenus teatud määral SNMP-d toetada (õigemini teenust tagav programm toetab SNMP-d), kuid pistikprogrammi abiga on avaliku teenuse jälgimine lihtsam ja oluliselt mugavam. Nagiose pistikprogrammide pakett on kätte saadav allikast [26], kuid igaüks võib neid programme ka ise programmeerida lähtudes oma vajadustest või soovist Nagiose tarkvara arendada.

Nagiose pistikprogrammide paketis olevate programmide ülevaadet ja juhendeid saab näha allika [41] alt, kus on selgitatud, milleks kindlat pistikprogrammi kasutada saab ja mis on selle programmi võimalused teenuse või seadme jälgimisel. Siinkohal oleks hea välja tuua mõned teenused ja nende jälgimiseks mõeldud Nagiose pistikprogrammid ning seejärel tutvustada natuke lähemalt ühte teenuse jälgimise programmi. Näiteks domeeninimede teenust kontrollib programm *check_dns*, failiedastusteenust kontrollib programm *check_ftp*, veebiteenust kontrollib *check_http*, e-posti teenust kontrollivad *check_imap, check_pop* ja *check_smtp*, ajateenust kontrollivad *check_ntp, check_ntp_peer* ja *check_ntp_time* ning turvalist kaughaldusteenust kontrollib *check_ssh*. Kõikide teenuste kontrolli mõte on selles, et Nagiose serverist proovitakse luua vastavat teenust pakkuvasse serverisse ühendus veendumaks, et avalik teenus töötab. Lisaks on võimalik teenuse jälgimisel kontrollida teatud parameetreid ning eeldada teenuse puhul kindlaid vastuseid- sellega tehakse näiteks kindlaks teenuse õigsuse vastavus, vaadatakse kas kindlale kasutajale on teenus kätte saadav ja veendutakse, et teenus vastab soovitud informatsiooniga piisavalt kiiresti. Lähemalt vaatleks siinkohal pistikprogrammi

check_http. Juba nime järgi saab eeldada, et tegemist on veebiteenuse kontrollimiseks mõeldud programmiga. Juhul kui veebiteenus töötab on *check_http*-ga veebiserverile tehtava päringu vastus järgnev: HTTP OK: HTTP/1.1 200 OK. Lisaks lihtsalt teenuse töösoleku kontrollile on pistikprogramm võimeline kontrollima ka turvalist veebiteenust (HTTPS), jälgima teenuse ümbersuundumist, otsima kindlaid sõnesid (*strings*) ja regulaaravaldisi (*regular expressions*), kontrollima ühenduseks loodavat aega, teavitama sertifikaadi aegumiseni jäänud päevadest ja muid olulisi parameetreid, mis võiks võrguhalduse spetsialisti huvitada.

5.2 Privaatsete teenuste jälgimine

Privaatsete teenuste jälgimise all on silmas peetud Windows operatsioonisüsteemide teenuste ja komponentide seiret. Kuna Windowsi tarkvara on tänapäeval väga laialdaselt kasutuses, siis leidub kindlasti ettevõtte võrguhaldust korraldavate inimeste hulgas neid, kes sooviksid jälgida servereid ja tööjaamu, kuhu on paigaldatud Windowsi operatsioonisüsteem. Selle jaoks on Nagioses kasutusel pistikprogramm *check_nrpe* (on olemas ka programm *check_nt*, kuid see on pigem iganenud ja ei soovitata enam kasutada [48]). Samas on vaja tööjaama paigaldada ka klientprogramm nimega NSClient++, et monitoorida Windowsi operatsioonisüsteemi teenuseid ja soovi korral ka komponente ning koormust. *Check_nrpe* pärib informatsiooni tööjaamas oleva NSClient++-i käest, mis omakorda küsib soovitud andmeid tööjaama käest ja edastab saadud tulemused Nagiose pistikprogrammile.

NSClient++ on lihtne, kuid võimekas seireprogramm, mis on mõeldud just Windowsi operatsioonisüsteemidele (toetatud on kõik versioonid alates NT 4.0-st kuni Server 2008-ni). See on programmeeritud Nagiose jaoks, kuid väikeste muudatustega saab seda arvatavasti integreerida iga seireprogrammiga. NSClient++ klientprogramm töötab Windowsi platvormi peal lihtsa teenusena, mille kaudu päritaksegi informatsiooni NSClient++-ga kaasas olevate moodulitega, mis omakorda sisaldavaid erinevaid pistikprogramme, või väliste skriptidega [42]. Pistikprogrammide abil on võimalik

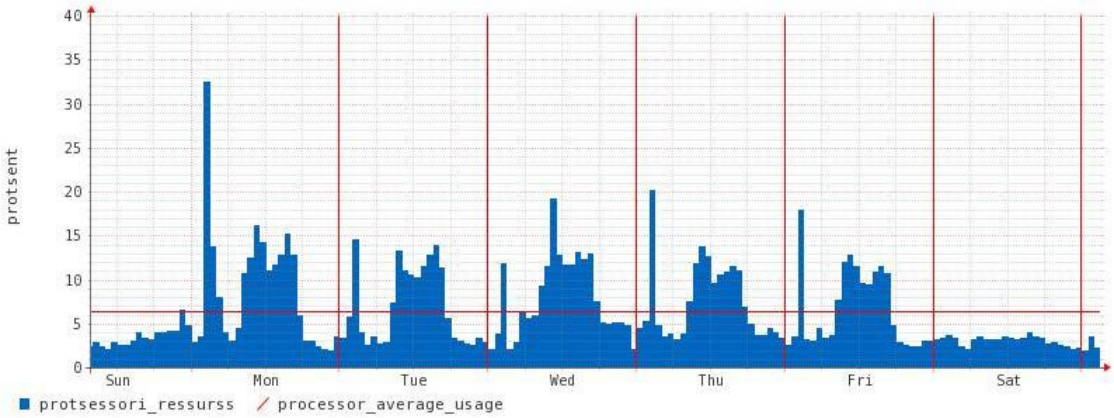
tööjaamast kätte saada informatsiooni protsessori, mälu ja kõvaketta kasutuse kohta ning uurida ka masina töösoleku aega. Seda informatsiooni saab kergesti kätte ka SNMP päringute abil. Mis teeb aga NSClient++ ja *check_nrpe* koostöö oluliseks Windowsi operatsioonisüsteemide jälgimisel on asjaolu, et saab jälgida teenuste tabelit, töötavaid protsesse ja sündmuste logi (Windowsi *event log*), vastavalt siis pistikprogrammidega *CheckServiceState, CheckProcState* ja *CheckEventLog*. Teenuste ja protsesside seirel on väga palju erinevaid võimalusi, alates sellest, et saab jälgida ühe kindla teenuse olekut lõpetades keerukamate kontrollidega, kus nõutakse kindla protsessi vähemalt viie eksemplari (*instance*) töötamist. Sündmuste logide puhul saab näiteks otsida kindlaid sõnu Windowsi logidesse tekkinud teadetest või pärida ainult neid kirjeid, mille olulisuse aste on tõrge (*error*). Teenuste tabeli, töötavate protsesside ja sündmuste logi jälgimine on kasulik ja äärmiselt vajalik, et näha kas Windowsi operatsioonisüsteemiga masinates töötavad kõik ärikriitilised programmid ja veendumaks, et nende töös ei esineks probleeme.

5.3 Raporteerimistarkvara

SNMP-ga seadmetest, serverites ja tööjaamadest informatsiooni hankimine võib osutuda väga vajalikuks tegevuseks, et ettevõtte infrastruktuurist ülevaadet saada. Võrguhaldusjaama tarkvara kasutamisel, näiteks Nagios, on võimalik kogu võrguseiret automatiseerida ja probleemide korral saada piisavalt kiiresti probleemse seadme kohta teade. Samas ei ole pea olema monitooringu ainukeseks eesmärgiks infrastruktuuris esinevate vigade jälgimine ja nendest teavitamine vaid tihti soovitakse seire abil saada ülevaadet ka seadmete igapäevasest koormusest ja normaalsest tööprotsessist. Selle ülevaate ehk siis raporti annavad väga ilmekalt graafikud, kust on lihtne vaadata ja saab kiiresti uurida seadmete koormust (ja ka muid näitajaid) nii päevase, nädalase, aastase kui ka ise valitud perioodi jooksul. Graafikuid tehakse eelkõige seadme koormusnäitajate kohta, kuid neid on võimalik luua ükskõik millele, kus päringu vastus on numbrilisel kujul (näiteks seadmesse sisse loginud kasutajate arvu kohta saab teha graafiku, kuid

pärides seadme asukoha kirjeldust, siis on vastus teksti kujul ja selle kohta pole võimalik mingit graafikut teha).

Programme graafikute tegemiseks on mitmeid ja Nagiose tarkvaraga integreerimiseks võib neist paljud leida allikast [43]. Magistritöö autoril endal on kogemus raporteerimisvahendiga NagiosGrapher, mis on kätte saadav allikast [44]. Nimetatud programm on tõesti väga hea abivahend graafikute põhjal raportite esitamiseks. Joonisel 12 on näidatud nädalane raport ühe ettevõtte tulemüüri protsessori kasutuse kohta. Antud graafiku pealt võib näha, et esmaspäevast reedeni (vertikaalsed punased jooned on päevade eraldajad ja horisontaalne punane joon näitab valitud perioodi kohta keskmist protsessori ressursi kasutamist) on tööpäeva jooksul protsessor 10-15 protsendi ulatuses töös, mis on täiesti normaalne kuna sel hetkel käivad töötajad internetis. Öösel ja nädalavahetustel jääb protsessori kasutus alla 5 protsendi, sest siis on väga vähe inimesi ettevõttes internetti kasutamas. Huvitav on aga graafikul see, et igal öösel umbes kella kahe ajal tarbitakse tulemüüri protsessorit 10-20 protsendi ulatuses ja esmaspäeva öösel isegi umbes 33 protsendi ulatuses. Selle põhjal saaks teha mitmeid järeldusi (näiteks toimuvad igal ööl rünnakud samal ajal või millegi pärast tuleb suurel hulgal kasutajaid sel hetkel kaugtöö kliendiga külge), kuid tõenäoliselt teostatakse öösel ehk kõige rahulikumal ajal andmete varundamist (*backup*) ettevõttest väljaspool asuva(te)le serveri(te)le.



Joonis 12. NagiosGrapheri graafik tulemüüri protsessori kasutuse kohta

Joonisel 12 oleva tulemüüri protsessor on piisavalt võimekas antud ettevõtte vajaduste täitmiseks kuna protsessori koormus on tööpäeva jooksul kui ka varundamise ajal küllaltki väike (isegi pool protsessori ressursist ei ole tipphetkel kasutuses nagu näha joonisel 12). Pigem on ettevõttes tulemüüri soetamisel mõeldud tuleviku peale ja ostetud võimekama riistvaraga seade. Samas võib olla graafikuid, kus on näha, et protsessori tarbimine on pidevalt 90 ja 100 protsendi vahel. Selline raport peaks andma IT juhile mõista, et nende tulemüür on liigselt üle koormatud ja tuleks mõelda uue soetamise peale, vastasel juhul hakkab tõenäoliselt peagi liikluse kvaliteet langema kuna tulemüür ei saa ühenduste haldamisega hakkama (uusi ühenduse soove ei aktsepteerita, olemasolevate ühenduste juures tekib järjest suurem paketikadu, sest saadetavaid ja vastuvõetavaid pakette ei suudeta töödelda ning neid ignoreeritakse jne). Selliste raportite esitamine oleks vägagi asjalik ja seetõttu peaks magistritöö edasiarendamisel kaaluma graafikute koostamise programmi integreerimist Nagiose tarkvaraga.

Kokkuvõte

Käesoleva magistritöö eesmärgiks oli koostada Tallinna Tehnikaülikooli infotehnoloogia teaduskonna raadio- ja sidetehnika instituudi telekommunikatsiooni õppetooli Andmeside kursuse jaoks SNMP labor, mis annaks tudengile ülevaate ja praktilise kogemuse võrguhalduse valdkonnast. Laboritöö koostamiseks uuris magistritöö autor esmalt võrguhalduse protokolli komponente ja toimimist. Seejärel vaadeldi viite maailmas populaarset võrguhaldusjaama tarkvara ning valiti nendest üks välja Andmeside kursuses tutvustamiseks. Laboritöö koostamiseks uuris ja analüüsis magistritöö autor ka erinevates maailma ülikoolides tehtavaid SNMP laboreid. Põhilises magistritöö osas, milleks on laboritöö ise, kirjeldas ja põhjendas autor erinevad ülesanded ja toimingud, mis tudeng peab läbima, et saada ülevaade SNMP-st ja valitud võrguhalduse tarkvarast. Lisaks on laboritööss mitmeid küsimusi, millele tudengil tuleb vastused ise otsida. Laboritöö koostati eelkõige magistritöö autori enda kogemuste põhjal, kuid samas lisati mõni ülesanne ka maailma ülikoolidest leitud informatsiooni põhjal. Labori koostamisega täitiski autor antud töö eesmärgi. Magistritöö viimases osas toob autor esile mõned võimalused, kuidas laboritööd saaks edasi arendada, et võrguhaldust täiustada.

SNMP on protokoll võrguhalduse jaoks, mille abil saab erinevatest SNMP-d toetavatest võrguseadmetest infot nii pärida kui ka seada. Nendeks seadmeteks võivad olla marsruuterid ja kommutaatorid kui ka printerid, modemid, erinevate operatsioonisüsteemidega arvutid jne. SNMP puhul vahetatakse informatsiooni kahte tüüpi üksuste vahel: üks on agent, mis töötab jälgitava võrguseadme peal ja teine on võrguhaldusjaam, mis viib läbi võrguseadmete seiret. Üksuste poolt vahetatav informatsioon asub objektidena haldusinfobaasis ning objektide struktuur määratakse haldusinfostruktuuriga. Objektide väärtuste pärimiseks agendist kasutab võrguhaldusjaam erinevaid protsesse. Lisaks on olemas protsess seadistuste muutmiseks võrguseadmes ja protsess agendi poolt võrguhaldusjaamale teate saatmiseks ilma eelneva päringuta. SNMP-st on kasutusel kolm versiooni, kus versioonid 1 ja 2 ei taga vahetatavale informatsioonile praktiliselt mingisugust turvalisust (ainult nõrga autentimise

kogukonnanime kujul, mille saab võrgus liikuvaid pakette jälgides lihtsasti teada). SNMP versioon 3 seevastu on turvaline tagades vahetatava informatsiooni autentimise ja krüpteerimise.

Magistritöö autor tutvustas viite erinevat võrguhaldusjaama. Nendeks olid Hewlett-Packard OpenView Operations, ServersCheck Monitoring Software, Zabbix, OpenNMS ja Nagios. Labori jaoks pidi ühe nendest välja valima ning arvestama sealjuures telekommunikatsiooni õppetooli õppejõudude poolt ette antud nõuetega. Põhiliseks nõudeks oli see, et võrguhaldusjaam peab olema vabavaraline ning töötama samuti vabavaralise süsteemi peal. Selle nõude tõttu langesid OpenView Operations, ServersCheck kohe välja kuna need on tasulised programmid. Ülejäänud kolme võrguhaldusjaama hulgast valis magistritöö autor laboris kasutamiseks välja Nagiose kuna omab kogemust sellega töötamisel ning leiab, et nimetatud programm on tudengi jaoks kergesti jälgitav ja lihtsasti arusaadav.

Enne Andmeside kursuse jaoks labori koostamist uuris magistritöö autor maailma ülikoolides läbi viidavaid SNMP laboreid, et ideid koguda ja vaadata, kuidas mujal võrguhalduse protokollist ülevaadet antakse. Autor kirjeldas uuritavaid laboreid, analüüsis nende sisu ja pani kirja missuguseid ülesandeid ja miks oleks hea kasutada ka magistritöö käigus loodavas laboritöös.

Käesoleva töö käigus koostas autor Andmeside kursuse jaoks labori, millega täitis magistritöö alguses püstitatud eesmärgi. Labor koosneb kahest põhilisest osast, millest üks on tudengi jaoks teoreetiline ja teine praktiline. Teoreetiline osa kestab orienteeruvalt 20 minutit ja selle eesmärgiks on anda tudengile lühiülevaade SNMP-st ja laboris toimuvast. Üldiselt on vaja enne laborisse tulekut tutvuda ka mõne materjaliga, et laboris toimuvast oleks põhimõtteline arusaam. Kuna enamasti keegi seda ei tee, siis otsustas magistritöö autor eeltöö läbi viia labori ajast, et kõigil oleks ülesandeid lahendades SNMP idee arusaadav. Teoreetilise osa väljundiks on ettekanne slaididena, mis on toodud lisas 1. Ülejäänud aeg on praktilise osa jaoks ehk erinevate harjutuste ja ülesannete lahendamine. Praktiline osa koosneb laboritöö juhendist, kus on kirjas kõik andmed

labori kohta, tegevuste juhend, aruande koostamiseks informatsioon, paketianalüsaatori kasutamise juhend ja kasulikud lingid, mis on abiks tudengitele küsimuste vastamisel. Magistritöö autor kirjeldas laboritöö juhendi koostamise juures kõik tudengi poolt läbi viidavad tegevused ning põhjendas, miks määratud tegevust on vaja ja mis see tudengile annab. Laboritöö oluliseks komponendiks on kolm ülesannet, mille tudeng peab ise erinevaid objekte uurides ja analüüsides ära lahendama. Lisaks on olulisemate tegevustena demonstreeritud SNMP turvalisust ja Nagiose abil automaatset võrguseiret. Pärast laboritöö tegemist peaks tudengitel olema piisavalt teadmisi SNMP-st ja selle rakendamise võimalustest.

Magistritöö viimases peatükis tõi autor välja mõningad laboritöö edasiarendamise võimalused. Kuna kogu laboris kasutatav tarkvara on vabavaraline, siis võib igaüks magistritöö käigus loodud laboritööd parendada, laborile midagi juurde lisada või lisaks teise võrguhalduse labori teha. Edasiarenduse variantidena pakkus magistritöö autor välja avalike teenuste (nende alla on mõeldud näiteks e-posti teenust, veebiteenust jne) ja privaatsete teenuste (nende all on mõeldud Windowsi operatsioonisüsteemi teenuseid) jälgimise, sest SNMP võimalused on nende valdkondade juures piiratud. Lisaks pakkus autor edasiarenduse võimalusena mõne raporteerimistarkvara integreerimist Nagiosega, et graafikute alusel oleks monitooringu tulemustest alati kiire ja hea ülevaade.

Simple Network Management Protocol lab assignment for telecommunication course

Master thesis

Märt Erik

Résumé

The purpose of this master thesis was to compose Simple Network Management Protocol (SNMP) lab assignment for telecommunication course. This topic was chosen and implemented by the author of present master thesis because familiarity about network management and SNMP and was also suggested by tutor as there is necessity for lab about network management, more specifically SNMP, that can be taught to telecommunication students. Before composing lab the author searched and investigated various SNMP lab assignments made in different universities all over the world. The aim for that was to seek some good ideas what can be used in Tallinn University of Technology SNMP lab assignment and to analyze those labs exercises applicability in master thesis lab. Some good ideas were found and those were added to telecommunication course lab assignment. Still basically everything in lab assignment was put together based on author own experience. Outcome of this master thesis contains lab that gives student overview about SNMP- how it works and what components does it include- and also practical experience with this protocol. This lab assignment gives also students an experience and demonstration with one Network Management Station (NMS) software called Nagios which monitors network appliances automatically and informs users when there are some problems in network. Present master thesis fulfilled its goal as given paper includes SNMP lab assignment for telecommunication course and also gives suggestions to supplement this master thesis in the future or even make one extra lab for managing different services in network.

Present master thesis first chapter covered SNMP overview. There were explained what units communicate (and how they communicate) with each other to exchange management information- these were agents and NMS-s. There were also described that management information base in agent holds different objects and those objects values can be queried by NMS and thus get useful network management information. In first chapter there were also discussed different processes that are used to query information and chapter's last part explained why SNMP versions 1 and 2 are not preferred compared to version 3 from security perspective. Second chapter introduced five popular and widely used NMS software which could be used in telecommunication course lab. One had to be chosen and based on Telecommunication chair professors requests and author's own preference Nagios was chosen. Master thesis third chapter contains investigated world universities SNMP lab assignments overview and analyzes. Fourth chapter can be thought as main chapter as it includes lab assignment description and exercises justification for telecommunication course. Lab assignment instruction itself is under appendix 2. The whole lab consists of two parts- first is theoretical which is presentation about SNMP and overview about lab and second is practical where students accomplish exercises based on lab assignment instruction. After lab all students must make report and send it to instructor. Master thesis last chapter gave some suggestions how lab assignment can be elaborated with services monitoring in the future as SNMP has not much capabilities on this field. Also reporting based on graphics were introduced as for example this helps to get convenient and fast overview about appliances resource consumption.

Kasutatud kirjandus:

1. IT ja sidetehnika seletav sõnaraamat. [WWW] www.vallaste.ee (06.02.2010)
2. Mauro, Douglas R., Schmidt, Kevin J. Essential SNMP. 2nd Edition. United States of America : O'Reilly Media, Inc., 2005.
3. SNMP over OSI. [WWW] http://www.ietf.org/rfc/rfc1418.txt (27.02.2010)
4. SNMP over AppleTalk. [WWW] http://www.ietf.org/rfc/rfc1419.txt (27.02.2010)
5. SNMP over IPX. [WWW] http://www.ietf.org/rfc/rfc1420.txt (27.02.2010)
6. SNMP - The Simple Network Management Protocol : Components and Architecture. [WWW] http://www3.rad.com/networks/applications/snmp/comp.htm (13.02.2010)
7. IT Expo and Conference. [WWW] http://www.interop.com (13.02.2010)
8. Protocol Operations for SNMP. [WWW] http://www.ietf.org/rfc/rfc3416.txt (15.02.2010)
9. Management Architecture. [WWW] http://www3.rad.com/networks/applications/snmp/img/image003.gif (13.02.2010)
10. MIB Tree Diagram. [WWW] http://www.manageengine.com/products/opmanager/images/mib-oid-tree.gif (22.02.2010)
11. Structure of Management Information – SMI. [WWW] http://traces.simpleweb.org/podcasts/tutorials/05-smi.m4v (22.02.2010)
12. SNMP - The Simple Network Management Protocol : Introduction to MIBs. [WWW] http://www3.rad.com/networks/applications/snmp/mibs.htm (23.02.2010)
13. Management Information Bases (MIBs) : Introduction. [WWW] http://www.simpleweb.org/w/images/b/b9/Tutorial_Slides_Mib-intro.pdf (23.02.2010)
14. Management Information Bases (MIBs) : MIB-II. [WWW] http://www.simpleweb.org/w/images/c/cc/Tutorial_Slides_Mib-2.pdf (23.02.2010)

15. MIB-II. [WWW] http://traces.simpleweb.org/podcasts/tutorials/08-mib2.m4v (23.02.2010)
16. SNMP - The Simple Network Management Protocol : SNMP Protocol Messages - Get, Set, and GetResponse. [WWW] http://www3.rad.com/networks/applications/snmp/getset.htm (25.02.2010)
17. Security in SNMPv3 versus SNMPv1 or v2c. [WWW] http://www.aethis.com/solutions/snmp_research/snmpv3_vs_wp.pdf (06.03.2010)
18. Simple Network Management Protocol (SNMP) : SNMP version 3. [WWW] http://www.simpleweb.org/w/images/c/c1/Tutorial_Slides_Snmpv3.pdf (08.03.2010)
19. Naumanis, E. Vabavaraline jälgimissüsteem IT-hooldusega tegelevale väikeettevõttele : diplomitöö. Tallinn, Eesti Infotehnoloogia Kolledž. 2007
20. Tulit, H. Monitooringu programmid: vabavaraline Nagios versus kommertsvaraline ServersCheck monitoring software 7.0 : diplomitöö. Tallinn, Eesti Infotehnoloogia Kolledž. 2007
21. Nagios. [WWW] http://kuutorvaja.eenet.ee/wiki/Nagios (27.03.2010)
22. InterMapper Pricing. [WWW] http://www.intermapper.com/buy-now/pricing/404-intermapper-pricing (27.03.2010)
23. Murdsalu, H. Monitooringu tarkvara Nagios integreerimine Tallink Grupp IT süsteemi : diplomitöö. Tallinn, Eesti Infotehnoloogia Kolledž. 2008
24. Comparison of network monitoring systems. [WWW] http://en.wikipedia.org/wiki/Comparison_of_network_monitoring_systems (27.03.2010)
25. Nagios. [WWW] http://sourceforge.net/projects/nagios/files/nagios-3.x/ (28.03.2010)
26. Nagios Plugin Development. [WWW] http://sourceforge.net/projects/nagiosplug/files/nagiosplug/ (28.03.2010)
27. Net-SNMP Download. [WWW] http://net-snmp.sourceforge.net/download.html (28.03.2010)
28. SNMP Trap Translator. [WWW] http://sourceforge.net/projects/snmptt/files/ (28.03.2010)

29. Nagios Documentation. [WWW] http://library.nagios.com/library/products/nagioscore/manuals/ (28.03.2010)
30. How to receive SNMP traps in Nagios. [WWW] http://xavier.dusart.free.fr/joomla/index.php/en/nagios/47-traps-snmp-dans-nagios (28.03.2010)
31. Lab 11.5.9: Installing SNMP. [WWW] ftp://81.111.95.43/e-books/Computer%20Engineering/Cisco/CNAP%20IT%20Essentials%20II%20-%20Network%20Operating%20Systems%202.0/PDF/lab_11_5_9.pdf (10.04.2010)
32. Lab 11.5.10: Configuring SNMP Security and Traps. [WWW] ftp://81.111.95.43/e-books/Computer%20Engineering/Cisco/CNAP%20IT%20Essentials%20II%20-%20Network%20Operating%20Systems%202.0/PDF/lab_11_5_10.pdf (10.04.2010)
33. Lab 5 - Performance Measurements and SNMP Monitoring. [WWW] http://www.itl.ohiou.edu/lab-perform.pdf (10.04.2010)
34. CS 556 - Networks II, Internet Teaching Lab (MCS B-24), Simple Network Mgmt Protocol (SNMP). [WWW] http://www.cs.bu.edu/fac/matta/Teaching/ITL/lab2-556S05.pdf (10.04.2010)
35. Do you know SNMP? [WWW] http://www.aims-conference.org/issnsm-2008/05-SNMP.pdf (11.04.2010)
36. Lab 9: Simple Network Management Protocol. [WWW] www.cs.utsa.edu/~korkmaz/teaching/mint/Lab9-1.doc (17.04.2010)
37. Õppeaine IRT0030 Andmeside koduleht. [WWW] http://www.lr.ttu.ee/wanlan/ (06.11.2010)
38. Download VMware Player. [WWW] http://downloads.vmware.com/d/info/desktop_downloads/vmware_player/3_0 (06.11.2010)
39. Download Wireshark. [WWW] http://www.wireshark.org/download.html (06.11.2010)

40. Arvutivõrgu halduse programmi kasutamine. [WWW] http://www.lr.ttu.ee/wanlan/labor/Andmeside_juhend_3.html (13.11.2010)
41. Nagios Plugins Manual. [WWW] http://nagiosplugins.org/man (27.11.2010)
42. About NSClient++. [WWW] http://nsclient.org/nscp/wiki/doc/about (27.11.2010)
43. Nagios Exchange - Graphing and Trending. [WWW] http://exchange.nagios.org/directory/Addons/Graphing-and-Trending (04.12.2010)
44. NagiosGrapher. [WWW] http://sourceforge.net/projects/nagiosgrapher/ (04.12.2010)
45. MIB Tree Diagram. [WWW] http://www.zohocorp.com.cn/manageengine/products/opmanager/images/mib-oid-tree.gif (04.12.2010)
46. InterMapper with Flows: Free Two-Week Evaluation. [WWW] http://www.intermapper.com/try-now (12.12.2010)
47. Cisco SFE1000P 8-Port 10/100 Ethernet Switch: PoE/Fanless. [WWW] http://www.cisco.com/en/US/prod/collateral/switches/ps5718/ps9967/ps9968/data_sheet_c78-501232.pdf (12.12.2010)
48. Using NSClient++ from nagios with check_nt. [WWW] http://nsclient.org/nscp/wiki/doc/usage/nagios/nsclient (12.12.2010)
49. Looking for HP OpenView? [WWW] https://h10078.www1.hp.com/cda/hpms/display/main/hpms_content.jsp?zn=bto&cp=1-10%5e36657_4000_100__&jumpid=go/openview (08.01.2011)
50. HP Operations Manager software. [WWW] https://h10078.www1.hp.com/cda/hpms/display/main/hpms_content.jsp?zn=bto&cp=1-11-15-28^1745_4000_100 (08.01.2011)
51. Monitoring Software. [WWW] http://files.serverscheck.net/brochures/2008_Monitoring_Software.pdf (08.01.2011)
52. Overview of Zabbix. [WWW] http://www.zabbix.com/documentation/1.8/manual/about (09.01.2011)
53. About OpenNMS. [WWW] http://www.opennms.org/about/ (09.01.2011)

54. OpenNMS Features List. [WWW] http://www.opennms.org/wiki/Features_List (09.01.2011)
55. Get OpenNMS. [WWW] http://www.opennms.org/get-opennms/ (09.01.2011)
56. OpenNMS. [WWW] http://www.opennms.org/ (09.01.2011)
57. Nagios Core 32bit Windows Installer. [WWW] http://exchange.nagios.org/directory/Distributions/Nagios-Core-32bit-Windows-Installer/details (09.01.2011)
58. SNMP Service (Windows). [WWW] http://msdn.microsoft.com/en-us/library/aa379100(v=VS.85).aspx (15.01.2011)
59. Versions of SNMP. [WWW] http://www.webnms.com/snmputilities/help/quick_tour/snmp_and_mib/snmpmib_versionssnmp.html (15.01.2011)
60. Popular Network Management Software in Comparison. [WWW] http://ipinfo.info/html/network_management_software.php (22.01.2011)
61. Setting up the WMI SNMP Environment. [WWW] http://msdn.microsoft.com/en-us/library/aa393621(v=vs.85).aspx (22.01.2011)
62. Aiko Pras. [WWW] http://www.vf.utwente.nl/~pras/ (22.01.2011)
63. Identity theft and cybercrime statistics in picture graphs. [WWW] http://blog.karenlodrick.com/2010/03/29/identity-theft-and-cybercrime-statistics-in-picture-graphs/ (22.01.2011)

LISA 1 DVD ketas

DVD ketta peal on järgmised failid:

* Nagios_32bit kaust, mis sisaldab laboritöö virtuaalmasinat.
* Laboritöö virtuaalmasina dokumentatsioon.
* Laboritöö teoreetilise osa ettekanne slaididena.

LISA 2 **Laboritöö juhend**

Võrguhaldus Simple Network Management Protocol baasil

Labor 3 juhend õppeaines Andmeside (IRT0030)

Märt Erik

Töö eesmärk

Tutvuda võrguhalduse jaoks kasutatava protokolliga SNMP ja seadistada ning tööle rakendada võrguhaldusjaama tarkvara Nagios.

Laboris kasutatakse arvuteid operatsioonisüsteemiga Windows 7, mille peal töötab virtuaalserver, kus on kasutuses Linuxi operatsioonisüsteemi distributsioon Fedora *release* 12. Nagiose tarkvara töötab Fedora peal. Laboris on kasutusel ka Cisco kommutaator SFE1000P 8-port 10/100 Ethernet Switch: PoE/ fanless.
Virtuaalserveri käivitamiseks kasutatakse programmi Vmware Player. Võrgust pakettide püüdmiseks on programm Wireshark.

Töö käik

1. Ettevalmistus

1.1 Kaustast C:\SNMP_labor kopeerida kaust Nagios_32bit töölauale (*desktop*ile). Programmide alt avada VMware Player ja valida *Open a Virtual Machine* ning

otsida töölauale kopeeritud kaustast *.vmx* laiendiga fail. VMware Playeris valida Nagios_32bit virtuaalmasin ja käivitada see valides *Play virtual machine.*

1.2 Virtuaalmasinasse sisse logimiseks on soovitav kasutada töölaual olevat Putty SSH klienti kuna selle kaudu on laborit teha mugavam (avada Putty programm, *Host Name (or IP address)* kasti kirjutada virtuaalmasina IP aadress, *Connection type* kohalt valida SSH ja vajutada *Open*). Võib kasutada ka VMware Playeri konsooli. Kasutajanimi ja parool on mõlemad „nagios".

1.3 Teha kindlaks arvuti IP aadress: *Start* > *cmd* > *ipconfig*. Panna see kirja.

1.4 Kontrollida arvuti ja Cisco kommutaatori (IP aadress 192.168.252.60) vahelist ühendust: *Start* > *cmd* > *ping* 192.168.252.60

1.4 Käivitada ja seadistada arvutis SNMP teenus. Teenuste tabeli nägemiseks valida *Start* > *services.msc* või *Computer* > hiire parem klõps > *Manage* > *Services and Application* > *Services*. SNMP *Service* peal hiire parem klõps ja *Start.* SNMP *Service Properties* alt valida sakk (*tab*) *Agent* ning seal valida *Service* alt ainult *Physical* ja omal valikul kirjutada ka kontakti ja asukoha kirjeldus. *Security* saki all lisada *Accepted community names* juurde kogukonnanimi *public* ja anda õigused *READ WRITE*. Lisaks valida *Accept SNMP packets from these hosts* ja lisada sinna oma virtuaalmasina IP aadress.

2. SNMP-ga informatsiooni pärimine

2.1 Virtuaalmasinast pärida *get* protsessi rakendades arvuti süsteemi kirjeldust: *snmpget –v 2c –c public arvuti_IP_aadress sysDescr.0*
Panna kirja terve saadud vastus.
Mis MIB-is asub päritav OID sysDescr?

2.2 Teha kindlaks objekti numbriline kuju. Selleks kasutada käsku: *snmptranslate –On MIB_fail::objekti_nimi*
Panna numbriline kuju kirja.

2.3 Käivitada Wireshark ja panna püüdma pakette, mis vahetatakse Cisco kommutaatoriga. Sooritada päring kommutaatori võrguliideste füüsiliste aadresside väljaselgitamiseks: *snmpwalk –v 2c –c public 192.168.252.60 ifPh*

Salvestada Wiresharki .pcap laiendiga fail.

Panna kirja ühe liidese kohta tulnud vastus.

Mitu erinevat võrguliidest on Cisco kommutaatoril?

Mis MIB-is asuvad päritavad objektid ja mis on objekti täielik nimetus?

Mitu päringut ja vastust kokku tehti võrguliideste aadresside teadasaamiseks (vaadata Wiresharkist)?

Mis SNMP protsessi rakendab käsk *snmpwalk*?

Miks on *snmwalk* käsk ebaratsionaalne suurte SNMP andmehulkade pärimiseks ja mis käsku/protsessi oleks mõistlikum kasutada? Selgitada oma vastust.

3. SNMP-ga informatsiooni muutmine

3.1 Avada võrgukaust \\Nagiose_virtuaalserveri_IP_aadress\MIB_failid. Kasutajanimi ja parool on mõlemad „nagios“.

3.2 Võtta lahti arvuti teenuste tabel ja avada SNMP *Service*'i atribuudid. Muuta *Agent* saki all olevad *Contact* ja *Location* väärtused virtuaalserveri konsoolist *snmpset* käsuga rakendades *set* protsessi.

Snmpset käsu kuju on järgmine:
snmpset -v 2c -c public seadme_IP_aadress OID OID_tüüp OID_uus_väärtus

OID ehk objekt kontakti ja asukoha kirjeldusele otsida SNMPv2-MIB-ist, mis asub eelnevalt avatud võrgukaustas. OID_tüübi leiab *snmpset* juhendist (trükkida *man snmpset*, väljumiseks vajutada tähte q). OID_uus_väärtus kontaktile on „tudeng@SNMPlaboris“ ja asukohale „arvuti_laboris“.

Veenduda, et *Contact* ja *Location* väärtused on *Agent* saki all muutunud (SNMP *Service Properties* aken sulgeda ja uuesti avada pärast *snmpset* käsu rakendamist).

Panna *snmpset* käsud ja neile antud vastused kirja.

4. SNMP turvalisus

4.1 Panna Wireshark jälgima võrguliiklust virtuaalmasina ja Cisco kommutaatori vahel.

4.2 Sooritada päring: *snmpget -v 2c -c public 192.168.252.60 sysName.0* ning salvestada Wiresharki ekraanipilt, kus on näha SNMP versioon ja päritavad andmed. Salvestada Wiresharki .pcap laiendiga fail.

4.3 Sooritada päring: *snmpget -v 3 -l authPriv -u snmpv3 -A esimeneparool -X esimeneparool 192.168.252.60 sysName.0* ning salvestada Wiresharki ekraanipilt, kus on näha SNMP versioon ja päritavad andmed. Salvestada Wiresharki .pcap laiendiga fail.

4.4 Mis on kahe päringu vahe ning miks peaks eelistama SNMP versiooni 3?
Mida tähendavad SNMPv3 päringul järgmised komponendid: *-l authPriv -u snmpv3 -A esimeneparool -X esimeneparool*?
Mis räsi- ja krüpteerimisalgoritme kasutab SNMP versioon 3 vaikimisi ja milliseid algoritme on üldse võimalik kasutada?

5. SNMP *trap*

5.1 Panna Wireshark jälgima võrguliiklust virtuaalmasina ja Cisco kommutaatori vahel.

5.2 Sooritada SNMP versiooniga 2 suvaline *snmpget* päring kommutaatori suunas kasutades kogukonnanimena midagi muud kui *public*.

5.3 Otsida Wiresharkist üles *trap* pakett/ paketid ning teha ekraanipilt, kus on näha *trap* OID kohta info numbrilisel ja sõnalisel kujul. Salvestada Wiresharki .pcap laiendiga fail.
Mitu *trap* teadet iga sündmusega saadetakse?
Selgitada püütud *trap*'i tähendust ja vajalikkust.

6. Ülesanded

Erinevate situatsioonide lahendamiseks kasutada ainult SNMP-d.

6.1 Luua Putty programmiga SSH ühendus Cisco kommutaatori suunas ja jätta ühendus aktiivseks. Lahendada järgnev situatsioon:

Võrguadministraator soovib muuta Cisco kommutaatori seadistusi. Ainuke võimalus seadet hallata on SSH kaudu. Pärast ühenduse loomist tahab administraator sisse logida, kuid saab teate: „Korraga saab seadet hallata ainult üks kasutaja“. Kuna ettevõttes on mitu inimesi, kes kommutaatorit haldavad, siis teeb järelikult keegi teine seadmes muudatusi või on lihtsalt kogemata ühenduse püsti unustanud. Administraatoril on vaja siiski kiiremas korras teha kommutaatoris mõned seadistused. Selle jaoks peab ta välja uurima, mis IP aadressi pealt on ühendus tulnud (SSH ühendusi võib olla ka mitu kommutaatori suunas, lihtsalt üks kasutaja on saanud sisse logida), et ta saaks paluda selle arvuti taga oleval inimesel välja logida ja oma muudatused kommutaatoris rakendada. Otsidagi välja SSH ühenduse loonud IP aadress või aadressid ning märkida need üles. Panna kirja terve käsk, millega tulemuseni jõuti ja ekraanipilt käsu tulemustest.

Vihjena olgu öeldud, et tulemuseni aitab jõuda kommutaatori TCP alampuu objektide uurimine.

6.2 Lahendada järgnev situatsioon:

Administraator näeb tulemüüri logikirjetes, et üks kindel arvuti ettevõttes on viirusega nakatunud ja saadab pidevalt erinevatele avalikele IP aadressidele mingisugust informatsiooni. See arvuti tuleb võimalikult kiiresti võrgust isoleerida. Administraator teab, missuguse võrguliidese küljes see kindel arvuti on ja seetõttu otsustab ta isoleerimiseks võrguliidese välja lülitada.

Lahenduse kontrollimiseks panna oma arvutist *ping*ima viirusega arvuti (*ping –t viirusega_arvuti_IP_aadress*) ja seejärel rakendada võrguliidest välja lülitav käsk. Veenduda, et enam ei saa viirusega arvutit *ping*ida ja lülitada kindlasti liides uuesti sisse ning veenduda, et viirusega arvuti on jälle kätte saadav.

Need, kelle virtuaalserverite IP aadressid on 192.168.252.61, 192.168.252.62 ja 192.168.252.63, kasutavad viirusega arvutit IP aadressiga 192.168.252.70, mis asub kommutaatoris liidese 7 küljes. Teistele on viirusega arvuti IP aadressiks 192.168.252.71 ja see asub kommutaatori võrguliidese 8 küljes.

Panna kirja käsud, millega kindel võrguliides välja lülitati ja uuesti sisse tagasi lülitati.

Vihjena olgu öeldud, et vajaliku informatsiooni leiab IF-MIB failist ja tuleb otsida liidese staatust haldavat objekti.

6.3 Leida järgnev informatsioon:

Administraatori soovib teada saada Cisco kommutaatori võrguliideseid läbinud liikluse hulka, et näha kui suur on koormus seadme erinevatel liidestel.

Panna kirja käsud ja ekraanipildina näidata saadud tulemused.

Millise liidese kaudu on kõige rohkem liiklust vastu võetud ja millise liidese kaudu kõige rohkem liiklust välja saadetud? Lisaks anda nende liideste edastatud liikluse maht väljendatuna megabittides.

7. Võrguseire Nagiose tarkvaraga

7.1 Avada võrgukaust \\Nagiose_virtuaalserveri_IP_aadress\Labor. Kasutajanimi ja parool on mõlemad „nagios“.

meiliaadressid.cfg failis panna nagios@ttu.ee asemele enda e-posti aadress ja salvestada fail.

7.2 Avada võrgukaustast fail labor.cfg. Sinna sisse määrata jälgimise alla oma arvuti ja Cisco kommutaator. Nagioses määratakse jälgitavad seadmed järgmisel kujul (semikooloni järel olev tekst on kommentaar ja seda pole vaja kirjutada):

```
define host{
        use             generic-host
        host_name       Seadme_nimi         ; oluline objektide määratlemisel
        alias           seadme_kirjeldus    ; suvaline kirjeldus
        address         seadme_IP_aadress
        contacts        admin
        }
```

Lisaks määrata kohe mõlema seadme juurde *trap*'ide vastuvõtmiseks vajalik seadistus:

```
define service {
        host_name               Seadme_nimi         ; sama, mis seadme juures
        use                     snmptrap-service
        contacts                admin
        }
```

7.3 Arvuti kohta seadistada vabal valikul jälgimisse kaks objekti MIB-ist *HOST-RESOURCES-MIB.txt.* Selgitada valitud objekte ja põhjendada miks just need objektid valiti. SNMP päringute teostamiseks on Nagioses kasutusel pistikprogramm (*plugin*) *check_snmp* ja objekti määratletakse järgmisel kujul:

```
define service{
        use                     generic-service
        host_name               Seadme_nimi  ; sama, mis seadme juures
        service_description     teenuse_kirjeldus
        check_command           check_snmp!-C kogukonnanimi objekt
        contacts                admin
        }
```

7.4 Kommutaatori kohta seadistada vabal valikul jälgimisse kaks objekti. Selgitada valitud objekte ja põhjendada miks just need objektid valiti. Kuna kommutaator toetab SNMPv3-e, siis peavad päringud turvaliselt toimuma. *Check_snmp* pistikprogrammi juhendi lugemiseks trükkida virtuaalmasina konsooli järgmine rida: */usr/local/nagios/libexec/check_snmp --help*

7.5 Kommutaatori kohta lisada veel ka turvaline päring objekti *ifOperStatus.9* kohta.

7.6 Nagiose teenusele teha virtuaalserveri konsoolist taaskäivitus käsuga: *service nagios restart.*

7.7 Veebibrauserisse trükkida https://Nagiose_virtuaalserveri_IP_aadress/nagios ja logida sisse. Kasutajanimi on „nagiosadmin“ ja parool on „nagios“.

7.8 Oodata paar minutit kuni kõikide objektide kohta on Nagiose lehel *Status* tulba all roheline tulemus ning teavitada sellest labori juhendajat. Juhendaja võtab kommutaatori üheksandast liidesest võrgukaabli välja ning paari minuti jooksul saadetakse trap'i ja liidese kohta e-posti aadressi peale teade (probleemide kohta). Juhendaja paneb võrgukaabli üheksandasse liidesesse tagasi ja selle peale saadetakse samuti paari minuti jooksul teated (probleemide paranemise kohta).

7.9 Kontrollida e-posti, kas Nagios on teavitused saatnud ning veenduda, et kõik jälgitavad objektid on korras ehk *Status* tulp roheline.

7.10 Panna kirja kõik Nagiose konfiguratsioonifaili lisatud definitsioonid ning Nagiose monitooringulehel teha ekraanipilt jälgitavatest asjadest. Panna kirja ka kõik Nagiose poolt saadetud e-posti kirjad.

8. Töö lõpetamine

8.1 Veenduda, et kõik vajalik on kirja pandud ja nõutud kohtadest ekraanipildid tehtud. Seejärel trükkida virtuaalmasina halduskonsoolis käsk *poweroff*, millega lülitatakse masin välja.

8.2 Kustutada arvuti töölaualt kaust Nagios_32bit.

8.3 Lülitada arvutis välja SNMP teenus tehes SNMP *Service*'i peal hiirega parema klõpsu ja valides *Stop*.

Aruanne

Aruanne koostada samade punktidena nagu on laboritöö juhend. Aruandesse kirjutada kogu informatsioon, mille kohta on öeldud panna kirja või märkida üles (käsud, päringud, aadressid, Nagiose seadistused ja situatsioonide lahendused) ning lisada tuleb kõik tehtud ekraanipildid. Vastata tuleb ka igale esitatud küsimusle. Samuti tuleb anda selgitused ja põhjendused kui nii on nõutud. Aruandesse ei ole vaja lisada Wiresharki .pcap laiendiga faile- need on tudengile abistavaks materjaliks aruande koostamisel.

Aruanne vormistada veebilehena, mis vastab W3C standarditele. Veebilehe standarditele vastavust saab kontrollida siit.

Wiresharki õpetus

Wireshark on avatud lähtekoodiga tarkvara arvutivõrkudes vigade otsimiseks, protokollide analüüsiks ja nende tundmaõppimiseks.

Tutvuda programmi Wireshark kasutamisega. Käivitada Wireshark. Pakettide püüdmist alustatakse *Capture->Interfaces*, valida õige võrguliides ja selle juures vajutada *Capture*. Püütakse kinni kõik paketid, mis arvuti juurde viivas kaablis liiguvad. Pakettide püüdmise lõpetamiseks vajutada nupule **Stop**.

Nüüd näidatakse akna ülesmises osas igal real ühte püütud paketti, keskmises osas ühe paketi dekodeeritud kuju alt üles OSI mudeli kihtide kaupa, akna alumises osas on see pakett 16-süsteemi arvude ja ASCII kujul. Et oleks näha ainult vajalikud paketid, saab neid filtreerida (*Filter*).

Kasutajate elu lihtsustamiseks on Wiresharki sisse ehitatud automaatne info dešifreerimine. Jälgige hoolikalt, mis tegelikult on paketi sees (kõige alumine akna osa Wiresharki aknas) ja mis on programmi poolt lisatud info.

Vältimaks kõikide pakettide püüdmist, võib enne pakettide püüdmist luua filtri, mis lubab läbi ainult sellele masinale saadetud ja edastatud pakette sh levisaate paketid. Filtri loomiseks enne pakettide püüdmist avage menüüst *Capture, Interfaces, Options* ja avanenud aknas kirjutada reale *Capture filter* järgmine *string*: **host *teie_masina_IP_aadress***. Pakettide püüdmise lõpetamiseks vajutada nupule **Stop**.

Soovitatav on enne vajalikud paketid välja valida ja siis need paketid salvestada faili. Pakette saab salvestada käsuga *Save* või *Save as* failidesse, mida saab avada Wiresharkiga. (Kogutud pakettide salvestamiseks tekst tüüpi faili on käsk *File->Export -> as "Plain Text" file*, seal aknas valida kas kõik, valitud või märgitud

paketid ja *All expanded*. Tekst tüüpi salvestamist teha siis kui on vaja näidata ühte paketti aruandes.)

Kasulikud lingid

SNMP

SNMP ülevaade: http://www3.rad.com/networks/applications/snmp/main.htm
Põhjalik ülevaade SNMP-st: http://www.simpleweb.org/

SNMP agendi paigaldamine Windows 2000, XP ja 2003 operatsioonisüsteemile: http://www.windowsreference.com/networking/install-snmp-in-windows-2000xp2003/
SNMP agendi paigaldamine Windows Vista ja 7 operatsioonisüsteemile: http://maximumpcguides.com/windows-7/install-simple-network-management-protocol-snmp/
SNMP agendi seadistamine Windowsi operatsioonisüsteemidel: http://support.microsoft.com/kb/324263
SNMP päringute teostamine, agendi paigaldamine ja seadistamine Linuxi operatsioonisüsteemidel: http://net-snmp.sourceforge.net/

Nagios

Nagiose koduleht: http://www.nagios.org/
Nagiose allalaadimislink: http://sourceforge.net/projects/nagios/files/
Nagiose dokumentatsioon: http://library.nagios.com/library/products/nagioscore/manuals/
Nagiosest eesti keeles: http://kuutorvaja.eenet.ee/wiki/Nagios

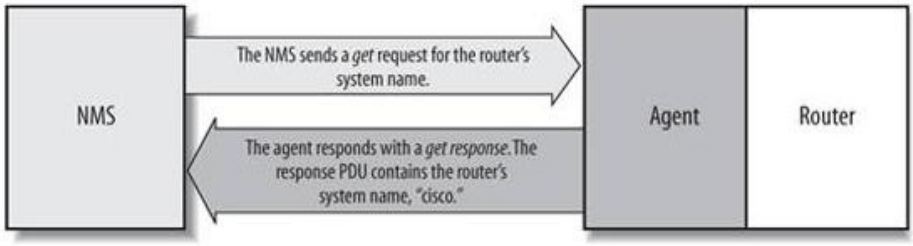
Labori ettekanne

Slaidid asuvad: http://www.lr.ttu.ee/wanlan/labor/Labor_3_ettekande_slaidid.pdf

LISA 3 SNMP protsessid

Get protsess

Get'i algatab NMS, mis saadab päringu agendile. Agent võtab päringu vastu ja töötleb seda vastavalt võimetele. Võib juhtuda, et mõni seade on suure koormus all (näiteks marsruuter) ja seetõttu ei suuda ta päringule vastata ning pakett kõrvaldatakse töötlusest (*drop*). Kui agendil õnnestub kogu soovitud info siiski koguda, saadab agent vastuse ehk *response* PDU tagasi NMS-ile, kus vastust töödeldakse. Kogu protsess on kujutatud joonisel 13. [2, lk 37]



Joonis 13. *Get* protsess [2, lk 38]

See, mida NMS täpselt agendiga seadmest otsib, määrataksegi *get* päringus objekti ja tema väärtuse paariga. *Get* päringu PDU formaat, kus otsitakse *sysContact* OID-d, näeb välja järgnevalt (kõik PDU formaadi näited on väljavõtted võrguliiklust analüüsivast tarkvarast Wireshark):

```
Simple Network Management Protocol
    Version: 2C (1)
    Community: public
    PDU type: GET (0)
    Request Id: 0x175f7f93
    Error Status: NO ERROR (0)
    Error Index: 0
    Object identifier 1: 1.3.6.1.2.1.1.4.0 (SNMPv2-
MIB::sysContact.0)
```

```
        Value: NULL
```

Nagu näha, siis on SNMP sõnumis määratud kõik väljad nii nagu need on kirjeldatud joonisel 6 (asub leheküljel 28) ja loomulikult ka täidetud väljade väärtused. PDU tüüp ongi *get*. *sysContact* OID on 1.3.6.1.2.1.1.4.0 ja samuti on sulgudes näidatud ka missuguses MIB moodulis antud OID asub. OID reaalne väärtus on *NULL* ehk siis *get* päringus OID tulemust ei ole, kuid see küsitaksegi agendilt.

Get päringu vastus ehk *response* PDU formaat on järgnev:

```
Simple Network Management Protocol
    Version: 2C (1)
    Community: public
    PDU type: RESPONSE (2)
    Request Id: 0x175f7f93
    Error Status: NO ERROR (0)
    Error Index: 0
    Object identifier 1: 1.3.6.1.2.1.1.4.0 (SNMPv2-
MIB::sysContact.0)
    Value: STRING: Root <root@localhost> (configure
/etc/snmp/snmp.local.conf)
```

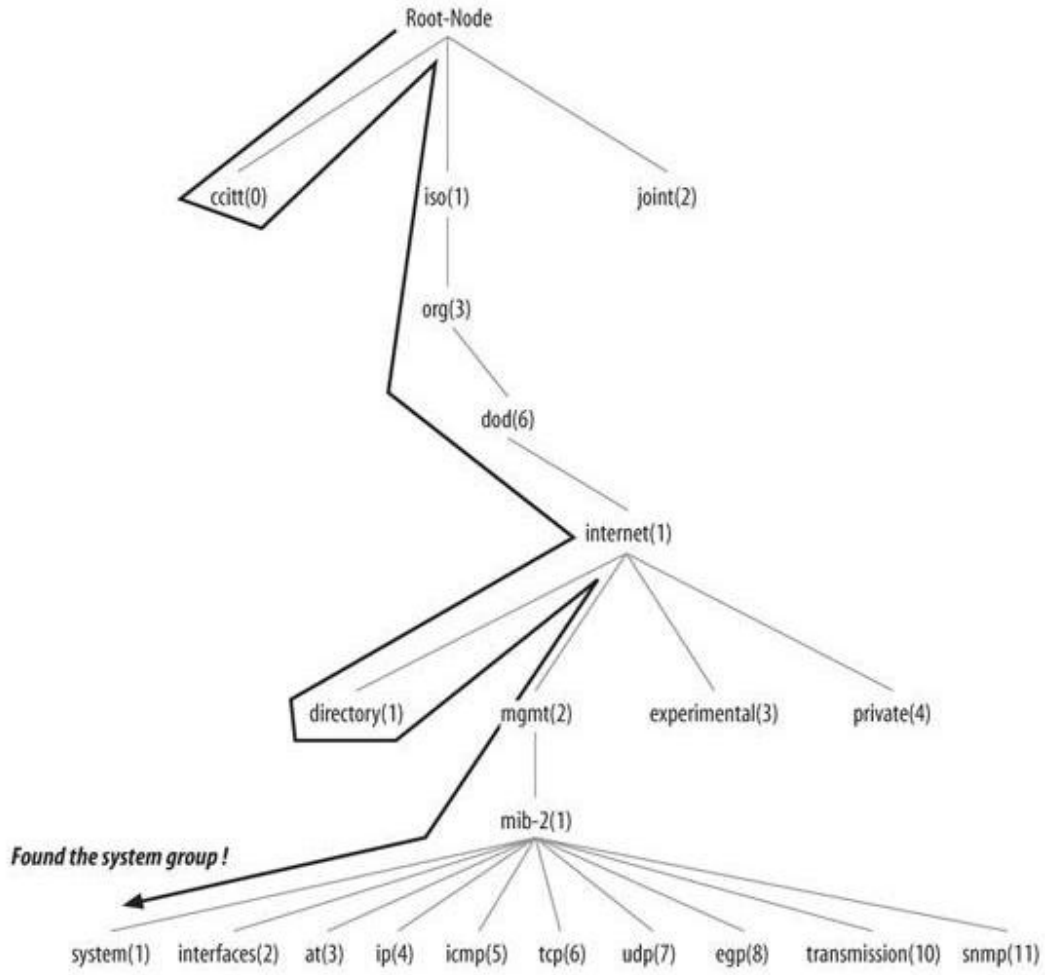
Päringu vastuses on näha, et PDU tüüp on *response* ja seekord on OID-l ka reaalne väärtus olemas, kus siis on näidatud, et seadme haldaja kontaktaadress on *root@localhost*. [2, lk 39]

Get on kasulik, kui on soov teada saada üks OID korraga. Selline objektide haldus võib olla ajakulukas ja tüütu. Siinkohal tuleb appi *getnext*, mis lubab seadmest otsida oluliselt rohkem kui ühe OID väärtuse. [2, lk 39]

Getnext protsess

Getnext protsess käigus täidetakse käskude jada, et välja otsida hulk väärtusi MIB-ist ehk siis iga MIB-i objekti jaoks, mida päritakse, tekitatakse eraldi *getnext* päring ja *response* vastus. *Getnext* käsk vaatab terve MIB-i või alampuu läbi sõnaraamatu struktuuri põhimõttel. Kuna OID on täisarvude jada, siis on agendil lihtne alustada objektipuu juurest ja liikuda puud mööda allapoole niikaua kuni soovitud OID on leitud. Kui NMS

saab agendilt vastuse tehtud *getnext* päringu kohta, siis edastatakse koheselt järgmine *getnext* käsk. NMS teeb seda nii kaua kuni agent saadab vastu veateate tähistamaks MIB-i lõppu ja ütlemaks, et rohkem objekte pärimiseks enam pole. [2, lk 43]



Joonis 14. *Getnext* protsess [2, lk 45]

Getnext protsessi kohta annab hea ülevaate joonis 14. Selle joonise puhul teostatakse *getnext* protsessi ning infot kogutakse *system*(1) rühma kohta. Info saamiseks *system* rühma kohta (mille OID on 1.3.6.1.2.1.1) alustatakse objektipuu juurest ja liigutakse allapoole. Iga sõlme juures vaadatakse kõige madalama numbriga alampuu sõlme. Seega, alustades juursõlmest, vaadatakse esmalt ccitt-d. Sellel sõlmel ei ole ühtegi alamsõlme ning liigutakse edasi iso sõlme juurde. Kuna iso sõlmel on alamsõlm, siis liigutakse edasi

org sõlme juurde. Protsess jätkub niikaua kuni jõutakse *system* sõlme juurde. Kuna iga alampuu koosneb suurenevatest täisarvudest (näiteks ccitt(0) iso(1) joint(2)), siis ei ole agendil probleeme käia selline puu struktuur läbi ja jõuda välja *system*(1) rühmani. Kui seda jada jätkata, siis järgmine on *system*.1, *system*.2 ja kõik teised objektid *system* rühmas. Edasi liigutaks *interfaces*(2) rühma juurde, aga antud näites soovitakse ainult *system* rühma infot. *Getnext* PDU formaat kahe esimese päringu ja vastuse puhul näeb antud näite puhul välja järgnev:

esimene *getnext* päring

```
Simple Network Management Protocol
    Version: 2C (1)
    Community: public
    PDU type: GET-NEXT (1)
    Request Id: 0x75cb182f
    Error Status: NO ERROR (0)
    Error Index: 0
    Object identifier 1: 1.3.6.1.2.1 (SNMPv2-SMI::mib-2)
    Value: NULL
```

päringu vastus

```
Simple Network Management Protocol
    Version: 2C (1)
    Community: public
    PDU type: RESPONSE (2)
    Request Id: 0x75cb182f
    Error Status: NO ERROR (0)
    Error Index: 0
    Object identifier 1: 1.3.6.1.2.1.1.1.0 (SNMPv2-
MIB::sysDescr.0)
    Value: STRING: Linux mailworks.guarded.net 2.4.21-4.EL
```

getnext teine päring

```
Simple Network Management Protocol
    Version: 2C (1)
    Community: public
    PDU type: GET-NEXT (1)
    Request Id: 0x75cb1830
    Error Status: NO ERROR (0)
    Error Index: 0
```

```
        Object identifier 1: 1.3.6.1.2.1.1.1.0 (SNMPv2-
    MIB::sysDescr.0)
        Value: NULL
```

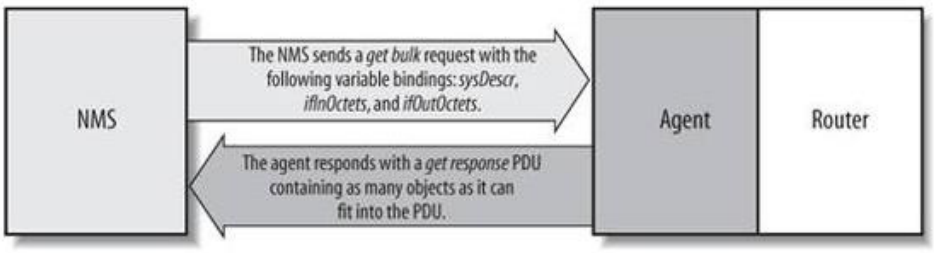
ning teisele päringule tulnud vastus

```
    Simple Network Management Protocol
        Version: 2C (1)
        Community: public
        PDU type: RESPONSE (2)
        Request Id: 0x75cb1830
        Error Status: NO ERROR (0)
        Error Index: 0
        Object identifier 1: 1.3.6.1.2.1.1.2.0 (SNMPv2
    -MIB::sysObjectID.0)
        Value: OID: SNMPv2-SMI::enterprises.8072.3.2.10
```

ja nii edasi kuni on kätte saadud kogu *system* rühma objektide väärtused. Päringutest on näha, et *getnext*'iga saadetakse agendile päring mingi OID kohta ja agent annab vastuseks sellest OID-st järgneva OID koos tema väärtusega. [2, lk 44]

Getbulk protsess

Getbulk on kasutuses SNMP versioonide 2 ning 3 juures ning see protsess lubab NMS-il pärida määratud skalaaride ja/ või tabeli osa ühe korraga. *Get* protsess võib küll üritada pärida rohkem kui ühe objekti, kuid sõnumi suurused on piiratud agendi võimalustega. Kui agent ei suuda tagastada kõikidele päringutele vastuseid, siis tagastatakse vea sõnum ilma andmeteta. *Getbulk* protsess seevastu laseb agendil saata nii palju informatsiooni kui võimalik ning see tähendab, et mittetäielikud vastused on lubatud. SNMP sõnumisse tuleb lisada kaks välja, et läbi viia *getbulk* protsessi: *nonrepeaters* ja *max-repetitions*. *Nonrepeaters* ütleb *getbulk* käsule, et esimesed N objekti saab otsida lihtsa *getnext* protsessiga. *Max-repetitions* ütleb *getbulk* käsule, et üritataks kuni M *getnext* protsessi, et otsida ülejäänud objekte. [2, lk 53]



Joonis 15. *Getbulk* protsess [2, lk 54]

Joonisel 15 on kujutatud *getbulk* protsessi näidet. Selle puhul päritakse kolme objekti: *sysDescr, ifInOctets* ja *ifOutOctets*. Objekti ja väärtuste paari koguhulk, mida päritakse, on antud valemiga M * R + N, kus N on *nonrepeaters* arv (ehk skalaarobjektid, mis antud juhul on 1, sest ainuke skalaarobjekt selles päringus on *sysDescr*), M on *max-repetetions* (antud juhul on see pandud suvaliselt 3) ja R on mitteskalaarsete objektide arv päringus (antud juhul on see 2, sest skalaarobjektid ei ole *ifInOctets* ja *ifOutOctets*). Arvestades numbreid näitest tuleb päritavate objekti ja väärtuse paaride koguhulgaks 3 * 2 + 1 = 7 ja vastus antud näite päringu peale näeks välja järgmine:

```
system.sysDescr.0 = " Linux snort 2.4.7-10 #1 Thu Sep 6 17:27:27
EDT 2001 i686 unknown "
interfaces.ifTable.ifEntry.ifInOctets.1 = 70840
interfaces.ifTable.ifEntry.ifOutOctets.1 = 70840
interfaces.ifTable.ifEntry.ifInOctets.2 = 143548020
interfaces.ifTable.ifEntry.ifOutOctets.2 = 111725152
interfaces.ifTable.ifEntry.ifInOctets.3 = 0
interfaces.ifTable.ifEntry.ifOutOctets.3 = 0
```

PDU formaat *getbulk* päringu puhul, kus küsitakse infot *sysDescr, sysContact* kohta ning *nonrepeaters* on määratud 1 ja *max-repetitions* 2, näeks välja alljärgnev:

```
Simple Network Management Protocol
    Version: 2C (1)
    Community: public
    PDU type: GETBULK (5)
    Request Id: 0x0f15c607
    Non-repeaters: 1
    Max repetitions: 2
    Object identifier 1: 1.3.6.1.2.1.1.1 (SNMPv2-MIB::sysDescr)
```

```
        Value: NULL
        Object identifier 2: 1.3.6.1.2.1.1.4 (SNMPv2-MIB::sysContact)
        Value: NULL
```

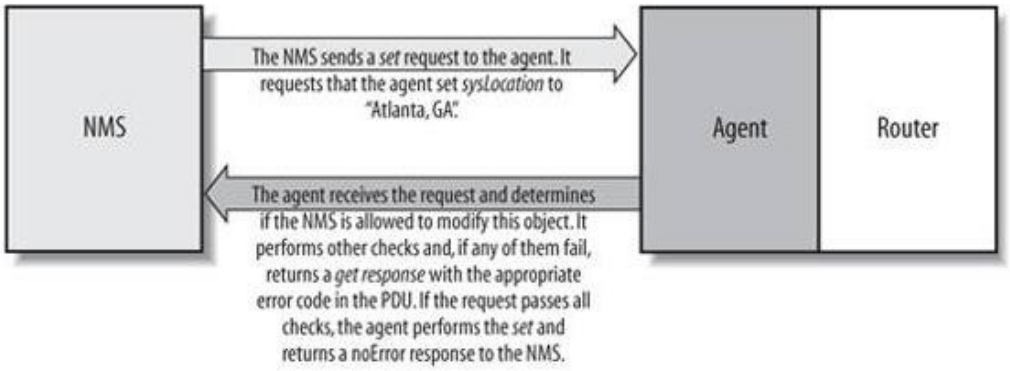
ja päringule saadetav vastus oleks:

```
Simple Network Management Protocol
    Version: 2C (1)
    Community: public
    PDU type: RESPONSE (2)
    Request Id: 0x0f15c607
    Error Status: NO ERROR (0)
    Error Index: 0
    Object identifier 1: 1.3.6.1.2.1.1.1.0 (SNMPv2-
MIB::sysDescr.0)
    Value: STRING: Linux mailworks.guarded.net 2.4.21-4.EL #1 Fri
Oct 3 18:13:58 EDT 2003 i686
    Object identifier 2: 1.3.6.1.2.1.1.4.0 (SNMPv2-
MIB::sysContact.0)
    Value: STRING: "kjs@guarded.net"
    Object identifier 3: 1.3.6.1.2.1.1.5.0 (SNMPv2-MIB::sysName.0)
    Value: STRING: box
```

Sellisel juhul küsitaksegi *getnext* käsuga, arvestades *nonrepeaters* välja, esmalt süsteemi kirjeldust ja seejärel, arvestades *max-repetitions* välja, süsteemi kontakti ja süsteemi nime. [2, lk 53]

Set protsess

Set protsessi kasutatakse hallatava objekti väärtuse muutmiseks või tabelisse uue rea loomiseks. Objekte, mis on MIB-is määratud *read-write* (lugemise ja kirjutamise) õigustega, saab muuta ja lisada. NMS-ist on võimalik muuta ka korraga rohkem kui üks objekt, kuid kui ühe objekti väärtuse muutmine peaks ebaõnnestuma, siis ei muudeta kõikide objektide väärtusi. [2, lk 57]



Joonis 16. *Set* protsess [2, lk 57]

Joonisel 16 on näha *set* protsessi toimimist. Põhimõtteliselt on see sarnane juba eelnevalt vaadeldud protsessidega, kuid väga oluline vahe on selles, et *set* puhul muudetakse seadme konfiguratsiooni. Joonisel näha oleva näite puhul määratakse *sysLocation* objektile väärtus. Kui väärtuse panemine peaks ebaõnnestuma, siis antakse vastuses NMS-ile tagasi vea kood, kui aga probleeme ei ole, siis seadistatakse seadmes objektile väärtus ja teadvustatakse sellest ka võrguhaldusjaama.

Set käsu puhul on PDU formaat:

```
Simple Network Management Protocol
    Version: 2C (1)
    Community: private
    PDU type: SET (3)
    Request Id: 0x34df1dbe
    Error Status: NO ERROR (0)
    Error Index: 0
    Object identifier 1: 1.3.6.1.2.1.1.5.0 (SNMPv2-IB::sysName.0)
    Value: STRING: box
```

ja kui probleeme ei esine siis saadetakse NMS-ile tagasi kinnitus, et objektile on uus väärtus lisatud:

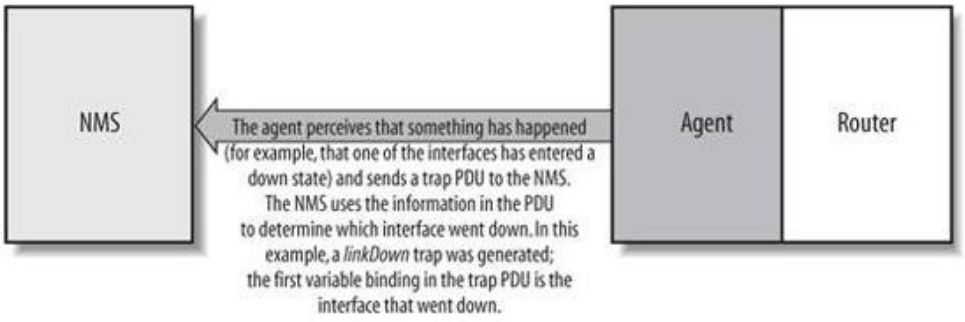
```
Simple Network Management Protocol
    Version: 2C (1)
    Community: private
    PDU type: RESPONSE (2)
    Request Id: 0x34df1dbe
```

```
Error Status: NO ERROR (0)
Error Index: 0
Object identifier 1: 1.3.6.1.2.1.1.5.0 (SNMPv2-IB::sysName.0)
Value: STRING: box
```
[2, lk 57]

Trap protsess

Trap on agendi viis öelda võrguhaldusjaamale, et midagi on seadmega juhtunud. *Trap* pärineb agendist ja saadetakse sihtkohta, mis on agendis seadistatud. Tavaliselt on *trap*'i sihtkoht NMS-i IP aadress. Mingit kinnitust NMS-i poolt *trap*'i peale ei saadeta, seega ei ole agendil infot, kas *trap* ikka sihtkohta jõudis. [2, lk 63]

Kui võrguhaldusjaam võtab *trap*'i vastu, siis peab ta oskama seda ka tõlgendada ehk NMS peab teadma, mida saadetud *trap* tähendab ja mis informatsiooni ta endaga kaasas kannab. Esmalt tehakse *trap* kindlaks tema üldise *trap*'i numbri järgi. Üldiseid *trap*'i numbreid on seitse (0-6). Üldine *trap* number 6 tähistab spetsiaalset kategooriat ettevõtte spetsiifiliste *trap*'ide jaoks. Need on *trap*'id, mis on määratud tootjate poolt ja ei kuulu ülejäänud kuue üldise *trap*'i kategooria alla. Ettevõtte spetsiifilised *trap*'id määratakse edasi kindlaks ettevõtte ID (OID ettevõtte alampuus kohas *iso.org.dod.internet.private.enterprises*) ja kindla *trap*'i numbri järgi (viimane määratakse ettevõtte poolt). Seega on objekti identifikaator ettevõtte spetsiifilise *trap*'i jaoks ettevõtte_OID.kindla_*trap*'i_number. Iga ettevõte võib vabalt määrata oma ettevõtte spetsiifilisi *trap*'e, kuid enne seda on nõue, et ettevõtte number peab olema registreeritud IANA juures. *Trap*'i saatmine on kujutatud joonisel 17. [2, lk 64]



Joonis 17. *Trap* protsess [2, lk 63]

Notification, *inform* ja *report* protsessid

Kõik kolm protsessi on kasutusel SNMP versioonidega 2 ja 3.

Kuna SNMP versiooni 1 *trap*'il on teistsugune PDU formaat kui *get*'il ja *set*'il, siis sooviti see ühtsele kujule viia ning SNMP versioon 2 defineeris *NOTIFICATION-TYPE*'i, mida kasutataksegi *notification* protsessi juures ja mille PDU formaat on samasugune kui *get* ja *set* PDU-d. [2, lk 67]

Inform protsess tagab teavitamise protseduuri, mille käigus edastatakse kinnitus saadetud *trap*'ile. Kui *inform* teade saadetakse, siis vastuvõtja edastab saatjale vastuse ja ütleb, et teade on kätte saadud. Selle protsessi käitumine on sarnane *get* ja *set* päringutele. *Inform* puhul saab saata SNMP versiooni 2 ja 3 *trap*'e võrguhaldusjaamale ja siinkohal ongi oluline aru saada, et sellisel juhul teadvustatakse agenti kohale jõudnud *trap*'ist. [2, lk 69]

Report protsess määratleti SNMP versiooni 2 mustandi (*draft*) versioonis, kuid seda ei rakendatud. Nüüd on see SNMP versiooni 3 standardi osa. *Report* protsessi mõte on lubada SNMP mootoritel (*engine*) omavahel suhelda (põhiliselt probleemide teavitamiseks SNMP sõnumite töötlemisel). [2, lk 69]