Internet-pohjaisen verkon havaintotiedon visualisointi

Janne Redsven

Tampereen yliopisto
Informaatiotieteiden yksikkö
Vuorovaikutteinen teknologia
Pro gradu -tutkielma
Ohjaaja: Veikko Surakka
Kesäkuu 2014

Tampereen yliopisto
Informaatiotieteiden yksikkö
Vuorovaikutteinen teknologia
Janne Redsven: Internet-pohjaisen verkon havaintotiedon visualisointi
Pro gradu -tutkielma, 51 sivua, 5 liitesivua
Kesäkuu 2014

TIIVISTELMÄ

Tutkielman tavoite oli vertailla graafista ja tekstipohjaista käyttöliittymää Internet Protocol -pohjaisen verkon seurannassa ja havaintotiedon visualisoinnissa. Osallistujat (N = 10) käyttivät tekstipohjaista käyttöliittymää sekä erikseen tätä tutkimusta varten toteutettua prototyyppikäyttöliittymää. Prototyyppi toteutettiin D3-sovelluskehyksellä siten, että verkkolaitteet sekä niiden portit mallinnettiin värikoodattuina solmuina ja solmujen välisinä yhteyksinä. Osallistujat oli jaettu kahteen ryhmään sen perusteella, kuuluiko heidän päivittäiseen työhönsä tekstipohjaisen käyttöliittymän avulla tehty ylläpito vai ei.

Mittauksen tuloksien perusteella graafinen käyttöliittymä oli tekstipohjaista käyttöliittymää hitaampi. Tekstipohjaisessa käyttöliittymässä tehtyjen kirjoitusvirheiden määrä oli merkittävästi suurempi verrattuna osoitusvirheiden määrään graafisessa käyttöliittymässä. Molemmat osallistujaryhmät pitivät kokonaisuudessaan graafista käyttöliittymää hankalampana käyttää. Haastattelujen perusteella graafinen käyttöliittymä sopii paremmin kokonaisuuksien hahmottamiseen ja tehtäviin, jotka liittyvät toisiinsa. Graafisen käyttöliittymän kehitystä kannattaneekin jatkaa saadun palautteen pohjalta.

Avainsanat ja -sanonnat: tietoverkko, seuranta, visualisointi, D3.

Sisällys

1. Johdanto

Yhä useampi päivittäin käytetty palvelu on riippuvainen tietoverkosta joko suoraan tai välillisesti. Paikallisesti asennettu palvelu tai sovellus saattaa toiminnassaan hyödyntää esimerkiksi tietokantaa, verkkolevyä tai muita tietoverkon palveluja kuten verkkoon liitettyä tulostinta. Tällöin ongelmat tietoverkon osissa saattavat näyttäytyä yllättävänä oireiluna käyttäjälle. Helppokäyttöisyys ei auta, jos sovellus ei toimi. Sovellus menettää toimimattomuutensa myötä käyttäjänsä. Siksi tietoverkon tilan seuranta on tärkeää.

Tietoverkon tilan jatkuva seuranta ei ole itsestään selvää. Seurannan vahvuus – tai sen puute – näyttäytyykin vasta silloin, kun seurannan avulla saatavalle tiedolle olisi käyttöä. Seurannan voi ajatella tähtäävän sujuvan käyttäjäkokemuksen ylläpitoon eli siihen, että verkon saatavuusongelmiin on reagoitu jo ennen kuin ne näkyvät käyttäjille.

Käytännön tarvetta seurannalle havainnollistetaan kahdella käyttötapausesimerkillä. Ensimmäisessä esimerkissä tietoliikenneverkon ylläpitäjä voisi olla kiinnostunut runkoverkon kuormituksesta ja haluaa yhdellä silmäyksellä nähdä tietyllä aikavälillä päivittyvän kuormitustilanteen. Yhdessä verkkotopologiaa havainnollistavan kuvan kanssa on mahdollista nimetä oleellisia yhteysvälejä ja verkkolaitteita, ja tätä kautta seurata niiden kautta kulkevia liikennemääriä.

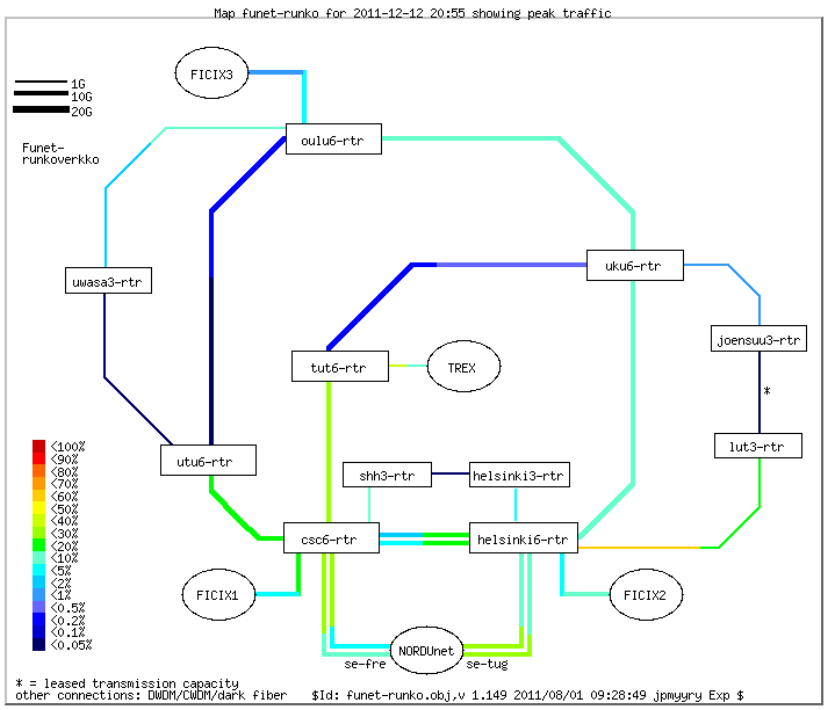
Eräs yksinkertainen sovellutusesimerkki tällaisesta on Finnish University and Research Network -tietoverkon (Funet) ”sääkartta” (ks. kuva 1). Kuvasta nähdään esimerkiksi, että Helsingin (helsinki6-rtr) ja Lappeenrannan (lut3-rtr) välisen suoran yhteyden, jonka tiedonsiirtonopeus on 1 gigabitti sekunnissa, hetkellinen kuormitus (peak traffic) on 50–60 prosenttia. Tämä tieto on olennainen, sillä jos käyttöaste lähentelee sataa prosenttia, esimerkiksi palvelunestohyökkäyksen tai muutoin suuren käyttöasteen vuoksi, mainittua yhteysväliä pitkin kulkeva verkkoliikenne kärsii jo merkittävää haittaa.

Yhteysväliä voi tutkia tarkemminkin: osoitettaessa sääkartalla yhteysväliä helsinki6-rtr–lut3-rtr valvontajärjestelmän kautta on mahdollista tarkastella graafisesti esimerkiksi mainitun yhteysvälin prosentuaalista kuormitusta (ks. kuva 2). Tästä kuvasta nähdään muun muassa, että tarkasteluhetken 50–60 prosentin kuormitus on ”ulospäin” helsinki6-rtr:iin nähden ja kuormitus ”sisään” on noin 20 prosenttia, eli Lappeenrannan suuntaan on enemmän liikennettä. Yhteysvälin kuormitus on myös kertaalleen ollut lähellä sataa prosenttia ja tämä voi, tilanteesta riippuen, antaa aihetta selvittää kuormituksen syitä. Kuormituspiikki voi myös selittää esimerkiksi muita samaan aikaan havaittuja verkkoon liittyviä ongelmia.

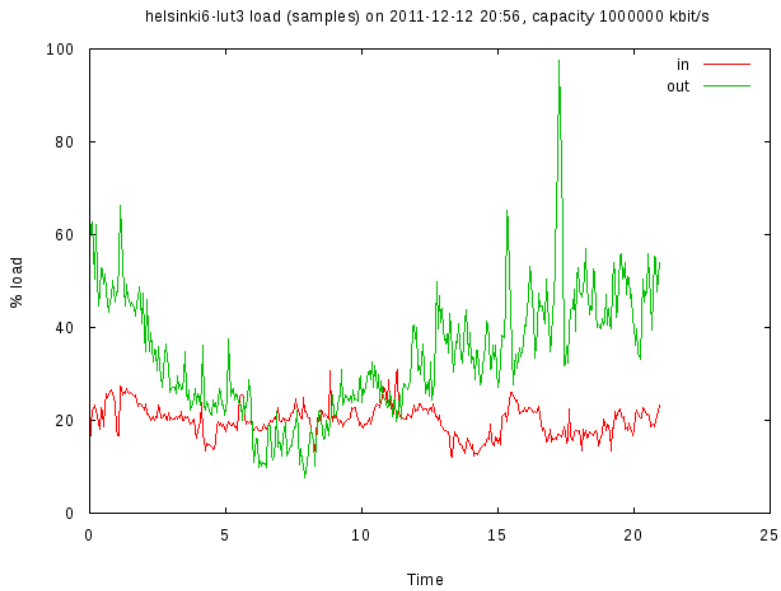
Vastaavalla tavalla voitaisiin tarkastella esimerkiksi Helsinki–Lappeenranta -yhteysvälin tietoliikennepakettien määriä tai yhteysvälillä ilmeneviä virheitä, kuten kehysvirheitä tai virheitä tämän yhteysvälin reitittimien porttitasolla.

Toisessa käyttötapausesimerkissä päähenkilönä on lähituen työntekijä. Asiakas ottaa yhteyttä ja kertoo, että työasemalla on vaikeuksia kytkeytyä lähiverkkoon tai liikennöinti tapahtuu hitaasti.

Lähitukihenkilö tarkistaa verkon seurantajärjestelmästä sen työasemaan liitetyn kytkinportin tilahistorian ja liikennemäärät selvittääkseen, onko vika verkossa vai esimerkiksi työaseman asetuksissa. Ideaalitapauksessa lähitukihenkilön käyttämässä seurantajärjestelmässä käyttöliittymä olisi ulkoasultaan sama, kuin ylläpitäjän käyttöliittymässä. Tällöin ylläpitäjä voisi seurata tilannetta runkoverkon tasolla (ks. kuva 1), kun taas lähituen työntekijälle tarkoituksenmukaista on porautua porttitasolle kuvan 2 kaltaiseen yksityiskohtaisempaan tietoon.



Kuva 1. Funet-runkoverkon ”sääkartta” (CSC, 2011)



Kuva 2. Liikennekuvaaja yhteysväliltä helsinki6-rtr–lut3-rtr (CSC, 2011).

On kenties hieman pessimististä pohtia mahdollisia häiriötilanteita, mutta asiaa voidaan tarkastella myös siitä näkökulmasta, että tieto ongelmista on ylläpidon kannalta tarkoituksenmukaisempaa saada omista järjestelmistä jo ennen käyttäjän tai teknisen tuen yhteydenottoa. Tietoa voidaan haluta kerätä vain tilastolliseen käyttöön. Toisaalta vain tilastollisesta näkökulmasta kerätty informaatio saattaa olla luonteeltaan liian epätarkkaa, eikä sen vuoksi auta virhetilanteissa. Kuitenkin virhetilanteita varten kerätty tieto voidaan tuonnempana havaita merkitykselliseksi tilastolliseltakin kannalta. Eri käyttäjäryhmien tiedontarpeet voivat luonnollisesti myös limittyä tai olla täysin päinvastaisia, jolloin yhden käyttöliittymän toteutusmalli on olennainen osa toimivaa tietoverkon seurantaa. Mutta miten tiedetään, millainen käyttöliittymä palvelee esitettyjä tarpeita?

Tätä tutkielmaa varten on rakennettu Network Administration Visualized (NAV) -nimisen avoimen lähdekoodin seurantajärjestelmän verkkotopologiakuvaa muistuttava graafisen käyttöliittymän prototyyppi havaintotiedon visualisoimiseksi. Uutta käyttöliittymää kehiteltäessä huomio kohdistuu käytettävyyteen, jolla tarkoitetaan sitä, miten hyvin käyttäjä saavuttaa tavoitteensa, miten tehokkaasti tavoite saavutetaan ja miten tyytyväinen käyttäjä on saatuun tulokseen ja sitä edeltäneeseen vuorovaikutukseen. Käyttöliittymää ei voine pitää ainakaan täysin soveltuvana käyttötarkoitukseensa, mikäli käyttö rutiinitehtävissä on totuttua käyttöliittymää hitaampaa. Eräs tapa vertailla kehitettyä ja olemassa olevaa käyttöliittymää on mitata molemmilla käyttöliittymillä tehtävän suorittamiseen kuluvaa aikaa. Tämän tutkielman tapauksessa kehitettyä graafista käyttöliittymää verrataan verkkolaitteissa yleisesti käytettyyn, tekstipohjaiseen käyttöliittymään. Suoritusaikaan perustuva analyysi käyttöliittymien keskinäisistä eroista täydentyy vertailtaessa käyttäjätyytyväisyyttä käyttöliittymien välillä. Käyttäjätyytyväisyys liittyy tuotteen käyttöliittymän käyttöön liittyviin tunnekokemuksiin, ajatuksiin ja mielipiteisiin.

Aihepiirien osalta tämä tutkielma jakautuu johdannon jälkeen kahteen temaattiseen kokonaisuuteen. Näistä ensimmäinen koostuu luvuista 2 ja 3. Luvussa 2 esitellään yleisimpiä menetelmiä, joilla internet-pohjaisesta tietoverkosta voidaan kerätä havaintotietoa. Samalla hahmotellaan havaintotiedon lajeja. Luvussa 3 käydään läpi visualisoinnin perusteita, mittaustuloksia, erilaisia käyttöliittymiä visualisoinnin esittämiseen, visualisoinnin suunnitteluperusteita ja toteutukseen liittyviä suunnitteluvalintoja. Luvun lopuksi esitellään seurantajärjestelmä, jonka visualisointinäkymään tutkielman yhteydessä kehitetty visualisointiprototyyppi pohjautuu.

Tutkielman toinen aihekokonaisuus koostuu luvuista 4–6. Luvussa 4 esitellään visualisointiprototyyppi, prototyypin käytettävyysmittauksen menetelmä ja mittauksen toteutus. Luvussa 5 raportoidaan käytettävyysmittauksen tulokset, joita pohditaan tarkemmin luvussa 6. Tutkielman päättää luku 7, jossa luodaan katsaus tuloksiin ja pyritään luomaan yhteenveto tutkielman molemmista kokonaisuuksista.

Tässä tutkielmassa käytetään kaikelle kerätylle tiedolle nimitystä havaintotieto. Nimityksellä pyritään neutraaliuteen, sillä kerätty tieto itsessään on laadutonta ja vasta sen hyödyntäminen

seuranta- tai valvontatarkoituksessa muuttaa havaintotiedon seurantatiedoksi. Yhtä hyvin saman tiedon pohjalta voitaisiin luoda anonyymiä tilastotietoa. Kyse on siis siitä, miten ja millaiseen tarkoitukseen tietoa hyödynnetään.

Vastaavasti seurantajärjestelmä-sanan tilalla voisi hyvin käyttää sanaa valvontajärjestelmä, joka ehkä kuvaisi paremmin itse toimintaa. Molemmilla sanoilla on nyky-yhteiskunnassa hieman orwellilainen, negatiivinen merkitys. Tässä yhteydessä on ehkä syytä mainita, että laadukkaasti toteutetulla valvontajärjestelmällä voidaan osaltaan varmistaa verkon käyttäjien yksityisyyden suojaa esimerkiksi haittaliikenteen automaattisella havainnoinnilla. Toisaalta syötteiden ja tulosteiden jäljitysketjua (audit trail) toteuttava valvontajärjestelmä mahdollistaa yksittäisen tiedon käsittelyvaiheiden jälkikäteisen tarkistamisen. Näin myös ylläpitäjien vastuullinen toiminta voidaan tarvittaessa myöhemmin todentaa.

Aihepiiristään huolimatta tässä tutkielmassa pyritään välttämään ylimääräistä teknisyyttä. Tutkielma rajautuu käsittelemään Internet- tai lähiverkon havaintotiedon keruuseen käytettäviä työkaluja sekä niiden visualisointimahdollisuuksia. Lukijan kuitenkin oletetaan jossakin määrin hallitsevan esimerkiksi Internet Protocol (IP) -verkon toiminnan perusperiaatteita, koska niihin ei tämän tutkielman puitteissa ole mahdollista paneutua.

2. Havaintotiedon keruu

Tässä luvussa esitellään ensin hieman tarkemmin sitä, miksi havaintotietoa ylipäänsä kannattaisi kerätä. Pohdinnan jälkeen siirrytään tarkastelemaan ensin tekstipohjaista komentorivikäyttöä, joka muodostaa ylläpidon sekä samalla esittelee tutkielmassa käytettävän tekstipohjaisen käyttöliittymän peruselementit. Tekstipohjaisen käyttöliittymän jälkeen tutustutaan havaintotiedon keruumenetelmiin ja niihin liittyviin havaintotiedon lajeihin.

2.1. Miksi havaintotietoa pitäisi kerätä?

Propagandatutkija Harold Lasswell kiteytti 1940-luvulla viestinnän siirtomallin kaavassaan ”kuka sanoo mitä minkä kanavan välityksellä kenelle ja millä vaikutuksella” (ks. esim. Pietilä, Malmberg & Nordenstreng, 2004). Tämä viestinnän kaava sopinee eräänlaiseksi rungoksi myös tietoverkossa kulkevan liikenteen seurannalle: häiriötilanteessa on tärkeää tietää, mistä liikenne tulee (kuka sanoo), mitä reittiä tai yhteyttä käyttäen (minkä kanavan välityksellä), mihin liikenne menee (kenelle) ja mitä se aiheuttaa (millä vaikutuksella). Näihin kysymyksiin pyritään vastaamaan keräämällä havaintotietoa.

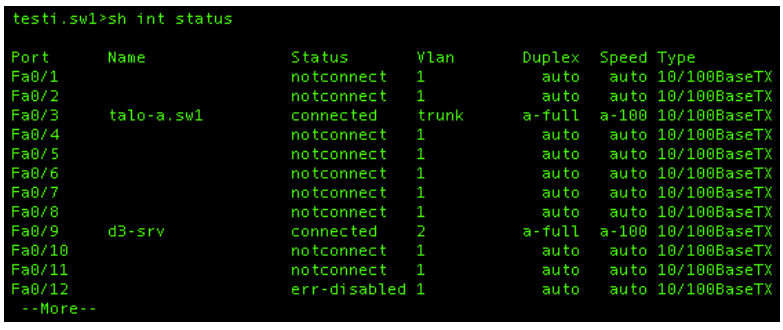
Vähemmän runollinen, joskin konkreettisempi, syy havaintotiedon keruulle on kirjattu lakiin. Sähköisen viestinnän tietosuojalaki (SVTSL) (16.6.2004/516, 19 §) esittää vaateen huolehtia siitä, että organisaation tietoverkko on turvallinen käyttää. Turvallisuudella tarkoitetaan sitä, että verkon tietoturvan tulee täyttää kolme vaatimusta: 1) luottamuksellisuus (confidentiality) eli tieto on vain valtuutettujen käyttäjien ulottuvilla, 2) eheys (integrity) eli tieto on vain valtuutettujen käyttäjien muokattavissa, eikä tieto muutu hallitsemattomasti esimerkiksi häiriötilanteissa ja 3) saatavuus (availability), jolla tarkoitetaan sitä, että tieto on valtuutettujen käyttäjien saatavilla, kun he sitä tarvitsevat (Valtiovarainministeriö, 2010, 29). Tämä on tietenkin itsestään selvää organisaatioissa, joissa käsitellään esimerkiksi henkilö-, viranomais- tai muita salassa pidettäviä tietoja. Kuitenkin laajat ongelmat tietoturvallisuuden ylläpidossa poikivat vähintään kielteistä julkisuutta, eikä henkilökohtaisen tiedon joutuminen vääriin käsiin ole kenenkään edun mukaista.

Verkon käyttäjät saattavat suhtautua tietoturvallisuuteen hieman leväperäisesti, sillä oman käsityksensä mukaan heillä ei ole mitään ”varastamisen arvoista”. Viestintäviraston yhteydessä toimiva Kyberturvallisuuskeskus (entinen CERT-FI) kuitenkin varoittaa muun muassa siitä, että hämärin keinoin hankittuja sähköpostiosoitteita voidaan käyttää esimerkiksi roskapostin lähettämiseen väärentämällä lähettäjäosoitteeksi varastettu sähköpostiosoite tai hyödyntämällä käyttäjän tunnusta sähköpostipalvelimelle (CERT-FI, 2010b). CERT-FI listaa joukon tyypillisimmin käyttäjältä varastettaviksi tiedoiksi muun muassa eri palvelujen kirjautumistiedot, luottokortti-, pankkitili- ja identiteettitiedot (CERT-FI, 2010a, 7–8). Tällä perusteella voidaan olettaa, että jokaisella käyttäjällä on vähintään ”verkko-minä”, identiteetti, jota voidaan helposti hyödyntää rikollisiin puuhiin.

Tässä tutkielmassa keskitytään tarkastelemaan tietoliikenneverkon toiminnasta saatavan tiedon saatavuuteen liittyviä kysymyksiä. Yleisenä suunnitteluperusteena voisi kuitenkin pitää tietoturvallisuuden ylläpitoa, eli myös luottamuksellisuuden ja eheyden säilyttämistä, jonka eräs muoto on näissä ilmenevien poikkeamien tunnistaminen ja niihin reagointi. Esimerkiksi viruksille tai muille haittaohjelmille saattaa olla tunnistettavissa ominainen tapa liikennöidä verkossa ja tällöin historiatiedon analysoinnilla voidaan kenties tuonnempana tarkastella niiden levinneisyyttä, pyrkiä jäljittämään alkuperää ja tunnistaa saastuneet työasemat tai verkkolaitteet. Historiatiedolla voidaan tarvittaessa tehdä myös rikosoikeudellista tutkintaa. Vaikka virka-apupyynnöt Suomen ulkopuolelle tuottavatkin valitettavan vähän tulosta, mahdollisimman tarkat tiedot auttavat tapauksen selvittelyssä. (CERT-FI, 2010a, 22.)

2.2. Tekstipohjainen komentorivi

Havaintotiedon keruumenetelmien tarkastelu on luontevinta aloittaa yksinkertaisimmasta menetelmästä, eli komentorivin käytöstä. Tekstipohjaisen komentorivin historiallinen tausta liittyy keskuskoneympäristöihin, jolloin yhteys päätteeltä keskuskoneelle oli tekstipohjainen ja perustui sarjaliikenteeseen (ks. esim. Mackay, Wright, Reynders, & Park, 2004). Toteutuksena sarjaliikenne on yhä käytössä ja verkkolaitteiden tapauksessa yksinkertaisesta, sarjaliikenteeseen perustuvasta yhteystoteutuksesta, käytetään nimeä konsoliyhteys (console access). Esimerkiksi asennusvaiheessa verkkolaitteeseen muodostetaan tekstipohjainen konsoliyhteys. Konsoliyhteys voidaan muodostaa esimerkiksi sarjakaapelilla ja pääte-emulaattorilla, joskin nyttemmin sarjakaapeli on korvautumassa tietokoneen USB-porttiin liitettävällä kaapelilla. Konsoliyhteyden avulla laite saadaan liitettyä verkkoon, jonka jälkeen sarjakaapelia ei tarvita, sillä yhteys konsoliin voidaan avata myös tietoliikenneverkon kautta. Sarjakaapelin avulla toteutettu konsoliyhteys toimii kuitenkin myös niissä tilanteissa, joissa verkkoyhteydet ovat muutoin poikki. Konsoliyhteys on siis tekstipohjainen yhteys verkkolaitteen käyttöjärjestelmään. Esimerkki Cisco IOS-käyttöjärjestelmästä on esitelty kuvassa 3, jossa tarkastellaan kytkimen porttien tilatietoa.



```
testi.sw1>sh int status

Port      Name               Status       Vlan       Duplex  Speed Type
Fa0/1                        notconnect   1            auto   auto 10/100BaseTX
Fa0/2                        notconnect   1            auto   auto 10/100BaseTX
Fa0/3     talo-a.sw1         connected    trunk      a-full  a-100 10/100BaseTX
Fa0/4                        notconnect   1            auto   auto 10/100BaseTX
Fa0/5                        notconnect   1            auto   auto 10/100BaseTX
Fa0/6                        notconnect   1            auto   auto 10/100BaseTX
Fa0/7                        notconnect   1            auto   auto 10/100BaseTX
Fa0/8                        notconnect   1            auto   auto 10/100BaseTX
Fa0/9     d3-srv             connected    2          a-full  a-100 10/100BaseTX
Fa0/10                       notconnect   1            auto   auto 10/100BaseTX
Fa0/11                       notconnect   1            auto   auto 10/100BaseTX
Fa0/12                       err-disabled 1            auto   auto 10/100BaseTX
 --More--
```

Kuva 3. Tekstipohjainen käyttöliittymä.

Yleisesti ottaen konsoliyhteyden avulla voidaan siis toteuttaa kaikki laitteeseen liittyvät hallinta- ja ylläpitotoimet, mukaan lukien havaintotiedon keruu. Konsolin käyttö on sinänsä yksinkertaista, mutta hankaluus piilee siinä, että eri verkkolaitevalmistajat käyttävät paikoin

erilaista komentokieltä laitteidensa käyttöjärjestelmissä – esimerkiksi Cisco IOS, Juniper JunOS tai HP ProCurve eroavat toisistaan käskykantansa ja tulosteiden osalta. Jos käytössä on usean eri valmistajan laitteita, saman tiedon hakeminen eri järjestelmistä saattaa siis edellyttää usean järjestelmäkohtaisen komennon muistamista.

Päivittäiskäytössä erilaiset komennot pysynevät paremmin mielessä. Tilannetta kuitenkin hankaloittaa se, että eri käyttöjärjestelmiä ei välttämättä erota pelkän komentokehotteen perusteella. Ylläpito ilman käyttöjärjestelmätietoa voikin alkaa käyttöjärjestelmätyypin selvityksellä, joka puolestaan saattaa vaatia omat, valmistajakohtaiset, käskynsä. Erityisesti epäsäännöllisessä käytössä tekstipohjainen käyttöliittymä saattaa tuntua monimutkaiselta ja hankalalta. Niinpä useat valmistajat ovatkin perusylläpitoa helpottaakseen lisänneet laitteen käyttöjärjestelmään mahdollisuuden muokata asetuksia selainkäyttöisen käyttöliittymän avulla. Valitettavasti selainpohjaiset käyttöliittymät poikkeavat toisistaan siinä missä tekstipohjaiset käyttöliittymätkin.

Havaintotiedon noutamiseen ja asetusten tekemiseen on kuitenkin olemassa standardoituja menetelmiä. Standardia noudattavien laitteiden kanssa voidaankin toimia ikään kuin samanlaisen rajapinnan avulla. Tällöin käyttöjärjestelmäsidonnaisuus saadaan osin häivytetyksi. Seuraavassa kohdassa käsitellään tarkemmin näitä standardeja ja niiden tuottamaa tietoa.

2.3. Standardit automatisoidun tiedonkeruun apuna

Havaintotietojen keruu voidaan ajatella kahden eri viestijän välillä tapahtuvana viestin vaihtona: asiakas (esimerkiksi seurantaohjelmisto) tiedustelee kohteelta (esimerkiksi verkkolaite tai palvelin) erilaisia ominaisuuksia tai attribuutteja. Viestintä voi tapahtua myös toiseen suuntaan eli verkkolaite tai palvelin voi lähettää viestejä seurantaohjelmistolle. Koska ominaisuuksia ja laitevalmistajia voi olla mielivaltainen määrä, on ollut välttämätöntä sopia jonkinlaisista viestinnän säännöistä. Tässä tutkielmassa näitä sääntöjä kutsutaan yhteyskäytännöiksi eli protokolliksi.

RFC 6632 (2012) esittelee Internet Protocol -pohjaisen verkon valvontaprotokollia, joista tähän tutkielmaan on valittu Simple Network Management Protocol (SNMP), SYSLOG ja IP flow information export (IPFIX) sekä Packet sampling (PSAMP). Tämä jaottelu koskee ainoastaan Internet Protocol (IP) -pohjaisia verkkoja ja on luonnollisestikin vain eräs jaottelu. Uudempia ja toisiin käyttöaiheisiin soveltuvia tekniikoita on olemassa, ja näistä muun muassa Subramanian, Gonsalves ja Rani (2010, 96–98) ovat laatineet koosteen. Tässä luvussa tarkastelun kohteena on kuitenkin tavanomaisten IP-tietoverkkojen toiminnallisuuden seuranta ja menetelmien esittely, ja siksi RFC 6632:ssa (2012) tehty jaottelu sopii riittävän yksinkertaisuutensa vuoksi. Samalla se kuitenkin antaa hyvän pohjan eri seurantamekanismien ymmärtämiseksi yleisellä tasolla.

Seuraavissa alakohdissa käydään läpi RFC 6632:ssa (2012) esitellyt protokollat ja pohditaan samalla niiden mahdollisia käyttötarkoituksia ja toiminnallisia eroja.

2.3.1. Simple Network Management Protocol

Internet Engineering Task Force (IETF) julkaisi vuonna 1988 ensimmäisen version Simple Network Management Protocol -standardista (SNMP) (RFC 1067, 1988). Stallings (1998) esittelee

SNMP-standardin kolme tärkeää ominaisuutta: 1) se määrittelee protokollan, jolla yksi tai useampi hallintajärjestelmä ja useat eri laitteet, eli agentit, voivat vaihtaa tietoa, 2) se määrittelee esitysmuodon välitettävälle ja laitteeseen tallennettavalle hallintatiedolle (management information) ja 3) se määrittelee useita yleiskäyttöisiä objekteja, joihin tieto tallennetaan. Objekteilla on yksilöllinen tunniste (OID, object identifier), jonka avulla niiden sisältöä voi lukea tai kirjoittaa.

Objektien hierarkkista kokonaisuutta kutsutaan nimellä Management Information Base (MIB). MIB-kokonaisuus on määritelty standardissa, joten toiminnallisuudeltaan samankaltaisissa järjestelmissä voi olettaa olevan samoja objektitunnisteita, jolloin järjestelmiä voi käsitellä samalla tavoin. Toisin sanoen eri valmistajien laitteet, jotka noudattavat standardia, antavat esimerkiksi tietoa tilastaan saman objektin kautta. Yleistäen objekti siis tarkoittaa muuttujaa, jolla on yksilöllinen tunniste ja jonka tyyppi on määritelty MIB:ssa. Valmistajilla on myös mahdollisuus laajentaa objektien kokoelmaa kirjoittamalla omia laajennoksiaan (vendor extensions). Tällöin valmistaja voi tarjota lisäobjekteja, jotka tarjoavat esimerkiksi tarkempia tietoja laitteen tilasta verkon ylläpitäjien hyödynnettäväksi. (Stallings, 1998, 37.)

Mainittakoon vielä, että SNMP-standardi mahdollistaa ominaisuuksien kyselyn lisäksi myös laitteiden tilan ja asetusten hallintaan liittyviä toimintoja: asetuksia voi muuttaa, eli hallintajärjestelmä voi tiedustella esimerkiksi muuttujan arvoa ja tarpeen mukaan muuttaa sen toiseksi. Laite voi tarvittaessa lähettää tilanmuutosviestin (trap) määritellylle vastaanottajalle. Ylläpitäjä voi määritellä laitekohtaisia ehtoja tilaviestien lähetykseen merkittävistä verkko-ongelmista. Tällaisia voisivat olla esimerkiksi tietyn raja-arvon ylittävä laitteiden välisten verkkoyhteyksien kuormitus tai laitteiden välisten yhteyksien katkeaminen. Tämän yksinkertaisen toiminnallisuuden ansiosta toiminnan seuranta on kevyttä, eikä hallintajärjestelmän tarvitse tältä osin käydä säännöllisesti läpi jokaista hallittavaa laitetta. Toisaalta myös seurantajärjestelmä voidaan asettaa reagoimaan eri tavoin tilaviesteihin, joten hälytykset saadaan lähes reaaliaikaisesti eteenpäin vaikkapa tekstiviestinä.

SNMP:n ensimmäinen versio, sittemmin kirjoitusasultaan SNMPv1, saavutti nopeasti suosiota ja laitevalmistajat alkoivat noudattaa sitä järjestelmissään. Ikävä kyllä SNMPv1:n kehityksessä ei huomioitu isojen tietomäärien kyselyä yhdellä kertaa ja sen tietoturvallisuus oli puutteellista. Muun muassa näitä ongelmia ratkomaan kehitettiin SNMPv2 (RFC 1441), joka standardoitiin vuonna 1993. (Stallings, 1998, 37) Uusi SNMPv2 paransikin mahdollisuutta kohdistaa kyselyjä useisiin objekteihin tai muuttujiin ja lisäsi joitakin uusia tietotyyppejä tiedon esittämiseen, vaikka parannusta tietoturvallisuuteen ei vielä tähän versioon saatukaan (Stallings, 2007, 760–769).

Esimerkiksi RFC 6632 (2012) toteaa, että SNMP:tä käytetään yleisesti virhetilanteiden ja suoritustehon seurantaan, sekä määriteltyjen toimintojen toiminnallisuuden seurantaan ja epäjatkuvuuskohtien havainnointiin (esimerkiksi käytettävyysaika nollaantuu). Laskurit myös mahdollistavat yhteismitallisen tilastotiedon keräämisen eri valmistajien laitteista. (RFC 6632, 2012, 8–10.) SNMP:n avulla siis voitaisiin rakentaa johdannossa ideoitu verkon ”sääkartta” (ks. kuva 1), jossa eri valmistajien laitteista kerätty tieto esitellään yhdellä ruudulla.

SNMP:n tarjoama havaintotieto ei kuitenkaan ole siinä mielessä kuvailevaa, että se ei kerro esimerkiksi verkkoliikenteen sisällöstä tai liikennöintiin käytettävistä protokollista mitään. Sen avulla ei myöskään voida selvittää esimerkiksi sitä, mihin kohdeosoitteeseen liikenne on matkalla tai mistä se tulee. Siksi myöskään kuvan 2 kuormituspiikin syistä ei voida sanoa kovin tarkasti mitään. Tämän vuoksi selvittelyä havaintotiedon keruumenetelmistä täytyy jatkaa.

2.3.2. SYSLOG

BSD SYSLOG (RFC 3164) on alun perin Unix-käyttöjärjestelmissä käytetty protokolla, joka välittää tila- tai lokiviestejä (log). SNMP:n tapaan BSD SYSLOG on yksinkertainen ja helppo käyttää, sillä perinteisesti BSD SYSLOG -viestillä on tarkoitettu tarkemmin määrittelemätöntä tekstipohjaista merkkijonoa ja kukin sovellus on voinut määritellä itse, millainen sisältö viestissä on. Ylläpitäjän on helppo tulkita tekstinä välitetyn viestin sisältö, mutta standardin puute on hankaloittanut merkittävästi viestien automaattista käsittelyä. Vuonna 2009 IETF standardoi uuden IETF SYSLOG -protokollan (RFC 5424, 2009) lisäämällä muun muassa kentät aikaleimalle, lähettävän isäntäkoneen nimelle, lähettävän sovelluksen nimelle ja sanoman tunnisteelle. Näiden lisäysten tarkoituksena on parantaa suodatusmahdollisuuksia, yhteensopivuutta ja vastaavuutta samankaltaisten sovellusten kesken. Lisäksi IETF on määritellyt IETF SYSLOG -protokollaan myös uusia ominaisuuksia, kuten tavan määritellä valmistajakohtaisia tietoelementtejä, mahdollisuuden käyttää yhteydellistä siirtotietä, salausta tai tilaviestien kuittauksen. Salauksella tarkoitetaan sitä, että lähetettävä tilaviesti salakirjoitetaan niin, ettei sitä ole mahdollista tulkita ilman salausavainta. Yhteydellinen siirtotie ja tilaviestien kuittaus on käsitelty jäljempänä. (RFC 5424, 2009; RFC 6632, 2012, 14–15.)

BSD SYSLOG -protokollan merkitys RFC 6632:n (2012) jaottelussa ei kenties tunnu ilmeiseltä, sillä myös SNMP tukee tilanmuutosviestien lähetystä. SNMP:n tilanmuutosviestit on kuitenkin tarkoitettu nimensä mukaisesti välittämään tietoa tapahtuneista muutoksista (joihin mahdollisesti toivotaan ylläpidon reagointia), joten ne eivät sinänsä sovellu esimerkiksi ilmoitusluontoisen tiedon välittämiseen. BSD SYSLOG -protokolla sen sijaan on suunniteltu kaikenlaisen tiedon välittämiseen, joten se täydentää havaintotiedon saatavuutta SNMP:n rinnalla. Tällöin esimerkiksi juuri syntyneestä häiriötilanteesta voitaisiin saada tieto verkkolaitteen lähettämän tilanmuutosviestin avulla. Häiriön syytä voitaisiin myöhemmin selvitellä tarkastelemalla BSD SYSLOG -protokollalla välitetyistä tilaviesteistä sitä, millaisia viestejä laite on lähettänyt ennen häiriötä.

Eräs käyttökelpoinen käyttöskenaario on myös tilanteet, joissa langattoman verkon tukiasema on vioittunut. Langattoman verkon luonteeseen kuuluu, että siihen langattomasti yhteydessä olevat päätelaitteet vaihtavat asiakkuutensa toiseen tukiasemaan esimerkiksi käyttäjän vaihtaessa paikkaa. Tällöin SYSLOG:n avulla langattoman verkon tukiasemat voivat lähettää tiedotusluonteisia tilaviestejä, joiden sisältönä on jonkinlainen päätelaitteen identifioiva tekijä ja siihen liittyvä

toimenpide, kuten tukiasemaan liittyminen tai poistuminen. Näin käyttäjien vihjeen perusteella päästään ongelmien jäljille.

Sekä BSD SYSLOG että SNMP on suunniteltu yksinkertaisiksi ja niihin tarpeisiin, joita suunnitteluhetkellä oli. Tämän vuoksi niissä on nykymittapuulla tarkasteltuna puutteita, joista merkittävimmät koskevat tietoturvallisuutta. SNMPv1 tai SNMPv2 eivät tue yhteyden salausta tai tietojen kyselijän käyttäjäpohjaista tunnistautumista. Tunnistautuminen SNMPv1 ja SNMPv2:ssa perustuu siihen, että kyselijän tulee tietää oikea yhteisön nimi (community), joka yksinkertaistettuna vastaa salasanaa ja kyselyn tulee saapua IP-osoitteesta, jolla on määritelty oikeus kysyä tietoja. Koska yhteys ei ole salattu, kaikki välitettävä tieto, ”salasana” mukaan lukien, voidaan poimia verkkoliikenteen seasta selväkielisenä. Luonnollisesti ongelmaa voidaan kiertää sillä, että esimerkiksi verkkolaitteet tyypillisesti sijaitsevat omassa niin sanotussa hallintaverkossaan, johon pääsy on rajoitettu ja tällöin myös riski tietojen päätymisestä vääriin käsiin pienenee.

Protokollan yhteydellisyydellä tarkoitetaan sitä, että kahden osapuolen välinen yhteys avataan ja suljetaan ennalta sovitulla tavalla ja tällöin molemmat osapuolet tietävät yhteyden tilan. Osapuolten toisilleen lähettämät paketit numeroidaan ja niille lasketaan tarkistussumma, jolloin vastaanottaja tietää niiden lähetysjärjestyksen ja kuittaa jokaisen saapuneen paketin. Paketteja voi myös eri syistä hukkua matkalle, ja tällöin tietyn ajan kuluessa vastaanottaja voi pyytää lähettämään uudestaan tietyn paketin, jolloin vastaanottaja voi varmistua siitä, että kaikki lähetetyt paketit ovat saapuneet perille. Tarkistussumman avulla vastaanottaja voi varmistua siitä, ettei paketin sisältö ole muuttunut matkalla. Eräs yleisimmistä IP-verkon protokollista on Transmission Control Protocol (TCP), joka on yhteydellinen sisältäen nämä mainitut toiminnallisuudet. (Stallings, 2007, 660–674.)

Yhteydettömässä protokollassa vastaanottoa ei mitenkään valmistella, eivätkä paketit sisällä järjestysnumeroa. Tällöin vastaanottaja ei voi tehdä päätelmiä pakettien järjestyksestä eikä siitä, ovatko kaikki lähettäjän tarkoittamat paketit saapuneet perille. Tarkistussumman avulla vastaanottaja voi tunnistaa jollakin tavalla vioittuneen paketin, mutta sillä ei kuitenkaan ole keinoa pyytää lähettäjää lähettämään pakettia uudelleen. IP-verkossa yleisin yhteydetön protokolla on User Datagram Protocol (UDP). Sekä TCP:tä että UDP:tä kutsutaan kuljetuskerroksen protokolliksi, sillä ne nimensä mukaisesti kuljettavat esimerkiksi sovelluksen lähettämän tiedon määränpäähänsä. (Stallings, 2007, 693–694).

Protokollaa, jonka tehtävänä on välittää tietynlaista tietoa, kuten SNMP välittää tietoa esimerkiksi laskureista tai BSD SYSLOG tilaviestejä, kutsutaan OSI-mallin mukaisesti sovelluskerroksen protokolliksi. Sovelluskerroksen protokolla puolestaan hyödyntää jotakin kuljetuskerroksen protokollaa viestin välitykseen. Myös muita kerroksia on, mutta niiden läpikäynti ei ole tämän tutkielman kannalta olennaista. Huomioitavaa on, että yhteydellisyys voidaan toteuttaa myös sovelluskerroksen protokollalla. Tällöin sovelluksen suunnittelussa huomioidaan yhteydellisyys. Tällöin sovelluksien välisessä yhteydettömässä viestiliikenteessä viestien sisällössä tuodaan ilmi se, että lähetetyt viestit ovat tulleet perille ja tarvittaessa muutoin todennetaan, että

yhteys sovellusten välillä on olemassa niin kauan, kun sitä tarvitaan. Esimerkiksi SNMP toimii yleisesti ottaen näin. (ks. esim. Stallings, 2007, 42-44; Stallings, 2007, 762–764.)

SNMP ja BSD SYSLOG -protokollat käyttävät UDP:ta kuljetuskerroksessa. Viestinnän osalta niiden eräs haaste on yhteydettömyys: lähettäjä ei saa kuittausta, kun BSD SYSLOG -tilaviesti tai SNMP-tilanmuutosviesti saapuu vastaanottajalle. Lähettäjällä ei siis ole mitään keinoa selvittää, saapuiko lähetetty viesti perille. Tällöin on mahdollista, että esimerkiksi tietoliikenneverkon häiriötilanteen vuoksi lähetetty SNMP-tilanmuutosviesti katoaa matkalle, eikä tieto häiriöstä koskaan saavuta ylläpitoa. SNMP-protokollan versio 2 mahdollistikin sovellustason yhteydellisyyden myös tilanmuutosviesteille (Stallings, 2007, 769). IETF SYSLOG -protokolla (RFC 5424, 2009) ja SNMP-protokollan versio 3 (RFC 3411, 2002) toivat parannusta mahdollistamalla tilaviestien yhteydellisen tiedonvälityksen (IETF SYSLOG), salatun tiedonsiirron ja SNMPv3:n tapauksessa myös käyttäjäkohtaisen tunnistautumisen sekä mahdollisuuden rajata pääsy vain ennalta määriteltyihin objekteihin.

Vaikka IETF SYSLOG esittelee joukon kaivattuja uudistuksia, se hyväksyttiin standardiksi vasta vuonna 2009, joka tekee siitä historiallisessa perspektiivissä sangen uuden protokollan. Tämän vuoksi kestänee vielä tovi, ennen kuin erilaiset laitteet ja sovellukset toteuttavat sen vaatimukset täysimääräisesti. Kestänee toinen tovi, ennen kuin sovellus- ja laitekanta uusiutuu niin, että kaikki tilaviestit voidaan siirtää IETF SYSLOG:lla. Tätä ennen esimerkiksi verkonvalvonnassa on varauduttava ottamaan tilaviestejä vastaan sekä BSD SYSLOG:lla että IETF SYSLOG:lla. Tämä puolestaan luo haasteita tulkinnalle, kun muistetaan, että ensin mainittu ei aseta mitään sääntöjä sille, miten tilaviesti pitäisi rakentaa.

Tässä vaiheessa on siis käsitelty tapa hankkia kvantitatiivista tietoa SNMP:llä esimerkiksi verkkoliikenteestä ja toisaalta erilaisten sovellusten tai laitteiden välittämiä tilaviestejä (BSD tai IETF SYSLOG). Näiden perusteella ei kuitenkaan voida vielä sanoa mitään siitä, millaista liikenne on: mistä se tulee, mihin se menee, kuinka paljon informaatiota siirtyy tai kuinka kauan yhteys on ollut muodostuneena. Tähän pulmaan yritetään seuraavaksi etsiä ratkaisua.

2.3.3. IP flow information export ja packet sampling

Tietoverkon IP-liikenne koostuu yhteyksistä (flow), jotka kulkevat verkon läpi kahden verkkoon liitetyn laitteen välillä tiettynä aikana. Tiettyyn yhteyteen, tai vuohon, kuuluvilla paketeilla on tästä syystä yhteisiä ominaisuuksia, jotka voidaan tunnistaa havaintopisteessä (observation point). Kyseessä ei ole TCP:n tai UDP:n tapaan yhteydellisestä tai yhteydettömästä protokollasta vaan siitä, että esimerkiksi kahden laitteen välillä kulkevat yhteydettömän(kin) protokollan viestit (esimerkiksi SNMP tai BSD SYSLOG) muodostavat havaintopisteessä havaittavan IP-pakettien vuon kahden laitteen välillä. Tämän vuon ominaisuuksia voidaan tarkemmin seurata. (RFC 3917, 2004.)

IP flow information export (IPFIX) on muun muassa RFC 3917:ssa (2004) määritelty protokolla, joka luo puitteet siihen, miten havaintotietoa yhteyksistä havaitaan, mitataan, viedään

(export) ja miten vietyä tietoa otetaan vastaan. Havaintotietoa yhteyksistä kerätään havaintopisteessä, joita voi olla yksi tai useampia. Havaintopisteenä voi toimia esimerkiksi reititin, jonka kautta liikenne kulkee. Havaintopisteessä muodostuu yhteystietueita (flow record), jotka sisältävät yhteyden ominaisuuksia, kuten lähde- ja kohdeosoitteen, sekä yhteydestä laskettuja suureita, kuten kaikkien yhteyden aikana liikennöineiden pakettien yhteenlasketun tavumäärään. Havaintopiste vastaa siitä, että yhteystietue luodaan, päivitetään, välitetään eteenpäin ja poistetaan tietyn ajan jälkeen, kun yhteys ei ole ollut aktiivinen. Yhteystietueet välitetään keräilijälle (collector), joka on tyypillisesti jokin ulkoinen järjestelmä, jossa esimerkiksi tietueet tallennetaan myöhempää käyttöä varten.

IPFIX:in avulla siis saadaan kerättyä tarkempaa tietoa siitä, millaista liikennettä tietoverkossa on. Kerätyistä tietueista voidaan luoda yhteenvetoja, joista selviää kunkin yhteyden kesto ja kellonaika, siirretty tavumäärä ja purskeisuus eli miten siirretty tavumäärä jakautuu koko yhteysajalle. Tietueista voidaan selvittää myös esimerkiksi yhteyksissä käytettyjä protokollia ja verkon keskeisiä palveluja. Tietueista haettavaa tietoa ei ole sinänsä ennalta määritelty, eli keräilijälle tallennetusta tiedosta voi tehdä millaisia hakuja hyvänsä. Eräs sovellusmahdollisuus onkin käyttää tietueiden sisältämää tietoa tunkeutumisen havainnointiin (intrusion detection), jossa voidaan etsiä esimerkiksi ennalta määrättyjä kohteita tai tietynlaista yhteyskäyttäytymistä. Tunkeutumisen havainnointiin toki liittyy joukko muitakin havainnointimenetelmiä, jotka eivät kuulu tämän tutkielman piiriin.

Havaintopisteet lienee järkevää sijoittaa sellaisiin tietoliikenneverkon kohtiin, joiden kautta kulkee mahdollisimman iso osa verkon liikenteestä. Tällaisia kohtia voisivat olla esimerkiksi internet-liityntäpiste ja keskeiset reitittimet. Liikenteen analysointi kuluttaa resursseja, joka voi esimerkiksi reitittimen tapauksessa heikentää kykyä suorittaa pääasiallista tehtävää eli liikenteen välittämistä. Valmistaja voi huomioida analysoinnin tehontarpeen liittämällä, tai myymällä, laitteistoonsa esimerkiksi erillisen prosessointimoduulin, joka on suunniteltu erityisesti liikenteen analysointiin. IPFIX kuitenkin mahdollistaa sen, ettei jokaista havaintopisteen läpi kulkevaa pakettia välitetä yhteyksien mittausprosessille, vaan paketeista voidaan valita tietty osajoukko (esimerkiksi joka sadas, jokainen TCP-paketti, satunnaisuuteen perustuva joukko jne.) ja käyttää vain tätä näytteistettyä osajoukkoa yhteystietueina. Myös tieto käytetystä näytteistyksestä välitetään keräilijälle yhteystietueessa. (RFC 3917, 2004.)

IPFIX ei aseta havaintopisteen mittausprosessille mitään vaatimuksia siitä, millaisia näytteistysmenetelmiä siinä tulisi olla toteutettuna. Tämän vuoksi vuonna 2002 alettiin kehitellä protokollaa, jonka tavoitteena oli muun muassa määritellä käytettävät näytteistysmenetelmät ja ne tiedot, jotka näytteistetyistä paketeista raportoidaan eteenpäin keräilijälle. Vuonna 2009 esiteltiin Packet Sampling -protokolla (PSAMP), joka määrittelee joukon näytteistysmenetelmiä. Näitä ovat muun muassa systemaattinen pakettien määrään perustuva näytteistys (esimerkiksi joka sadas), systemaattinen aikaan perustuva näytteistys (kuten pakettien määrään perustuva näytteistys, mutta näytteistetään paketteja aikavälein lukumäärän sijaan), erilaisia todennäköisyyksiin perustuvia

näytteistyksiä, paketin ominaisuuksien perusteella tehtävä näytteistys tai paketin sisällöstä laskettaviin tiivisteisiin (hash) perustuva näytteistys. (Claise & Wolter, 2007, 212) Näytteistysmenetelmien tarkempi esittely on kuitenkin rajattu tämän tutkielman ulkopuolelle.

IPFIX on siis yleinen yhteystietueiden vientiin suunniteltu protokolla, jonka vahvuus on välittää yhteyksiin liittyvää (usein kumulatiivista) tietoa. PSAMP taas määrittelee joukon menetelmiä, joilla voidaan näytteistää paketteja ja välittää tietoa kunkin yksittäisen paketin ominaisuuksista. Tällöin IPFIX ja PSAMP tukevat toisiaan: verkossa voidaan IPFIX:in avulla seurata yleisemmin IP-liikennettä ja PSAMP:in avulla valita tietty osajoukko liikenteestä tarkempaan analyysiin. Vaikka IPFIX ja PSAMP soveltuvat erilaisiin tarpeisiin, RFC 6632 (2012, 15–18) käsittelee niitä yhdessä. Syy on yksinkertainen, kuten Claise ja Wolter (2007, 213) toteavat: IPFIX ei tee eroa sen välillä, onko välitetyssä yhteystietuessa tietoa yhdestä vai useammasta paketista. IPFIX on nimittäin suunniteltu yleiskäyttöiseksi protokollaksi havaintotiedon vientiin, jolloin PSAMP:illa näytteistetty yksittäinen paketti voidaan ajatella IPFIX-protokollan näkökulmasta yksittäiseksi yhteystietueeksi. Tällöin PSAMP voi hyödyntää IPFIX:issa määriteltyä tapaa viedä yhteystietueet keräilijälle, jolloin myös sen toteutus muuttuu valmistajan näkökulmasta yksinkertaisemmaksi.

2.4. Keruumenetelmien tuottama havaintotieto

Tiedonkeruumielessä tässä luvussa esitellyt mekanismit antavat tuiki erilaista tietoa kukin: esimerkiksi alakohdassa 2.3.1 esitelty SNMP mahdollistaa hyvin laajan otannan, jolta tietoa voidaan kerätä. Alakohdassa 2.3.2 esitelty SYSLOG tukee SNMP:n avulla kerättyä tietoa tilaviestein. Tilaviestit kuitenkin kerätään tyypillisesti eri paikkaan kuin SNMP:llä kerätty havaintotieto, sillä tilaviestejä voidaan haluta vastaanottaa verkkolaitteiden lisäksi myös esimerkiksi Unix-pohjaista käyttöjärjestelmää käyttäviltä palvelimilta ja työasemilta.

Vaikka tilaviestit vastaanotettaisiinkin samaan paikkaan SNMP:llä kerätyn havaintotiedon kanssa, ongelmaksi muodostuu se, miten yhdistää helposti SNMP-havaintotieto ja tilaviestit. SNMP:llä kerätty tieto täytynee sijoittaa jonkinlaiselle aikajanalle ja etsiä tältä aikajanalta poimittuja aikaleimoja vastaavat merkinnät tilaviesteistä. Tällöin aikaleimojen tulee vastata mahdollisimman hyvin toisiaan, toisin sanoen laitteiden kellot tulisi pitää mahdollisimman hyvin ajassa. (Tähän on toki menetelmiä, esimerkiksi RFC 958:ssa määritelty Network Time Protocol, mutta niiden käsittely on tämän tutkielman aiheen ulkopuolella.) Sinällään tilaviestien tarkastelu pienellä aineistolla on suhteellisen yksinkertaista. Toisaalta jos SNMP:n havaintotiedon tarkasteluväli on oletuksellinen viisi minuuttia, aikahaarukka tarkasteltaville tilaviesteille on myös viisi minuuttia. Tarkasteltavien tilaviestien määrä voi kasvaa kohtuullisen suureksi, jos tilaviestejä saapuu esimerkiksi keskeiseltä verkkolaitteelta. Tällöin tilanmuutosviestit voivat auttaa paikallistamaan ongelman syntyhetkeä paremmin.

Siinä missä SNMP ja SYSLOG tarjoavat tietoa esimerkiksi yhteyksistä eri verkkolaitteiden liityntöjen (interface) välillä, alakohdassa 2.3.3 esitellyt IPFIX ja PSAMP syventävät SNMP:lla ja SYSLOG:lla kerättyä havaintotietoa tarjoamalla yksityiskohtaisempaa tietoa esimerkiksi tietyllä

hetkellä havaintopisteessä kulkevasta liikenteestä. IPFIX:in ja PSAMP:in avulla kerätty havaintotieto on myös luonteeltaan yksilöivää, sillä se paljastaa verkkoon liitettyjen laitteiden keskinäiset yhteydet. Sen vuoksi näiden protokollien avulla kerätyn havaintotiedon säilytykseen tulee kiinnittää erityistä huomiota. Havaintotietoa myös kertyy, riippuen näytteistystiheydestä, ripeästi, joten tallennusresurssien oikea mitoitus on tärkeää. Pakettien ja yhteyksien tarkempi tarkastelu siis lisää SNMP:n ja SYSLOG:n oheen vielä kolmannen kokonaisuuden. Tämä osaltaan lisää käytettävissä olevan havaintotiedon määrää suorassa suhteessa havaintopisteiden ja verkossa kulkevan liikenteen määrään.

Automatisointia voidaan siis hyödyntää monin tavoin havaintotiedon keruussa ja jatkojalostuksessa. Seuraavissa luvuissa tarkastellaan edellä kuvattujen menetelmien avulla kerättyä havaintotiedon visualisointia, visualisoinnin edellytyksiä ja vertaillaan graafisen käyttöliittymän avulla toteutettua tiedonhakua alakohdassa 2.2 esiteltyyn tekstipohjaiseen käyttöliittymään.

3. Havaintotiedon visualisoinnista

Tässä luvussa esitellään havaintotiedon visualisoinnin edellytyksiä. Aluksi kohdassa 3.1 käydään läpi visualisoinnin perusperiaatteita ja tähän liittyvässä kohdassa 3.2 käsitellään visualisointiin liittyviä tutkimustuloksia. Kohdassa 3.3 edetään visualisoinnin toteutusvaihtoehtoihin ja edelleen kohdassa 3.4 pohditaan mahdollisuuksiin hyödyntää www-selainta interaktiivisena käyttöliittymänä. Kohdassa 3.5 käsitellään joitakin seurantajärjestelmän suunnitteluperiaatteita, jonka jälkeen kohdassa 3.6 edetään verkkotopologian visualisointiin liittyviin käsitteisiin. Luvun lopuksi kohdassa 3.7 esitellään kahta erilaista tapaa toteuttaa seurantajärjestelmä.

3.1. Visualisointimantra ja tiedon koodaus

Tarve visualisoida on ollut olemassa koko ihmiskunnan historian ajan. Aiemmin tyypillinen käyttötarkoitus oli muun muassa opetus tai ajankohtaisten tapahtumien kirjaus. Nyttemmin visualisoinnin tarkoituksena on helpottaa laajojen verkkojen keskinäisten yhteyksien hahmottamista. Tällaisia verkkoja ovat esimerkiksi tietoliikenneverkot, sosiaaliset verkostot tai tekniset järjestelmät. (Kruja, Marks, Blair & Waters, 2002, 283–284.)

Interaktiivinen visualisointi poikkeaa perinteisestä visualisoinnista siinä mielessä, että käyttäjä voi vaikuttaa siihen, mitä visualisoidaan. Ben Shneiderman (1996) tiivisti visualisoinnin perusperiaatteet nyrkkisääntöön nimeltä visualisointimantra. Mantra kuuluu lyhykäisyydessään: *näytä ensin yleiskuva, lähennys ja suodatus, sitten yksityiskohdat kysynnän mukaan*. Mantran mukaan samasta tietosisällöstä muodostetusta visualisoinnista tulee olla muokattavissa erilaisia esityksiä käyttäjän tarpeista riippuen. Visualisoinnin tulisi siis muuttua graafisen sisällön lisäksi myös merkityksellisen sisällön osalta: yksityiskohtia tuodaan esiin ja poistetaan näkyviltä kulloisenkin tarpeen mukaan. Visualisointimantran keskeinen anti lienee se, että mantra ikään kuin luo ylätason toimintaperiaatteen visualisoinnille. Tehtävälistan toteuttava visualisointi ei ole sidottu mihinkään tiettyyn suunnitteluperiaatteeseen, vaan mahdollistaa omaehtoisen tutkimusmatkailun visualisoinnin kohteeksi valittuun tietosisältöön. (Shneiderman, 1996.)

Tiedon representaatiolla tarkoitetaan tiedon uudelleen esitystä (re-present). Esimerkiksi numeerinen havaintoarvo koodataan uudelleen, jolloin arvo voidaan esittää visuaalisesti. Representaatioon liittyy läheisesti kolme teemaa: tietotyyppi, tiedon kompleksisuus ja koodatun tiedon tulkinta. Tietotyypillä tarkoitetaan esimerkiksi muuttujien välisiä suhteita tai havaintotiedon muuttujien muodostamaa rakennetta. Tiedon kompleksisuudella tarkoitetaan sitä, kuinka moniulotteisena yksittäistä tietoelementtiä voi pitää. Esimerkiksi myytävään asuntoon voi liittyä useita määritteitä: huoneiden lukumäärä, valmistumisvuosi, yhtiövastike, lainaosuus, vesimaksun suuruus, kaupunginosa ja niin edelleen. Ongelma visualisoinnissa ei siten liity asuntojen lukumäärään, vaan siihen, miten erilaiset määreet saadaan koodattua visuaalisiksi elementeiksi. (Spence, 2007, 29–31.) Koodatun tiedon tulkinta puolestaan liittyy yhdessä havainnon kanssa kognitiivisiin prosesseihin, joiden lopputuloksena käsitys muodostuu (Spence, 2007, 27).

Tiedon koodaus voisi tässä asiayhteydessä tarkoittaa sitä, että esimerkiksi värien, symbolien tai viivanpaksuuksien avulla samaan visualisointiin voidaan yhdistellä eri tietolähteistä haettua tietoa. Esimerkkejä tästä voisi olla esimerkiksi verkkokytkimen tapauksessa muutos kuvakkeessa (ongelmatilanteet), väritys (onko portin tila kytketty vai ei, mahdollinen virhetilanne), kytkinporttiin kytkeytyneen laitteen ja kytkinportin välisen yhdysviivan paksuus (yhteysnopeus, esimerkiksi 10, 100 tai 1000 megabittiä sekunnissa) ja viivan täyteväriväri (liikennemäärän käyttämä kapasiteetti suhteessa maksimiin). Tämän lisäksi kohteisiin voidaan sisällyttää tarvittaessa haettavaa tietoa, joka esitetään esimerkiksi vietäessä osoitin kohteen päälle.

Visualisointimantra ja tiedon koodaus muodostavat teoreettisen pohjan visualisoinnille. Seuraavaksi tarkastellaan lyhyesti visualisointiin liittyvää tutkimusta, erilaisia tapoja tuottaa visualisointeja ja havainnollistetaan esimerkein syitä, joiden perusteella on päädytty tässä tutkielmassa toteutettuun visualisointiprototyyppiin.

3.2. Visualisoinnin mittaamisesta

Kohdassa 3.1 esitelty teoreettinen kehys ei sinällään puutu siihen, miten tehokkaasti visualisointi toimii tai onko visualisointi ylipäänsä aiempaa esitysmuotoa käyttökelpoisempi. Niinpä visualisoinnin teorian ohella on aiheellista tarkastella sitä, miten visualisointia voidaan mitata. Visualisoinnin voinee tulkita graafisen käyttöliittymän yhdeksi erikoistapaukseksi, jolloin visualisointeihin olisi mahdollista soveltaa käytettävyystutkimuksen menetelmiä. Tämän tutkielman tapauksessa vertailukohtana visualisoinnille oli olemassa oleva tekstipohjainen käyttöliittymä, joten esitellyt tutkimusmenetelmät ja -tulokset keskittyvätkin graafisen ja tekstipohjaisen käyttöliittymän väliseen vertailuun.

Eräs tapa asettaa käyttöliittymät vertailukelpoiseksi on mitata kummallakin käyttöliittymällä tiettyyn tehtävään kuluvaa suoritusaikaa. Sherry Koshman (2004) vertaili tiedonhakuun liittyvässä tutkimuksessaan tekstipohjaista ja graafista käyttöliittymää toisiinsa. Koshmanin tutkimuksessa mitattiin tehtäviin kuluvaa aikaa, onnistumisprosenttia, virheiden lukumäärää ja käyttäjätyytyväisyyttä. Osallistujat oli jaettu osaamistason perusteella noviiseihin ja ekspertteihin. Samankaltainen ajanmittaukseen perustuva asetelma on myös muilla (ks. esim. Staggers & Kobus, 2000; Whiteside, Jones, Levy & Wilson, 1985; Chen & Zhang, 2007).

Graafisen ja tekstipohjaisen käyttöliittymän suoritusaikavertailuissa ei tunnu olevan yksiselitteistä tulosta paremmuudesta. Esimerkiksi Koshman (2004) ei havainnut merkitsevää eroa suoritusajoissa käyttöliittymien välillä eri osallistujaryhmissä. Tilastollisesti merkitsevä ero suoritusajoissa käyttöliittymien välillä havaittiin, kun käyttöliittymän kautta saatujen hakutulosten määrää vaihdeltiin. Ero ilmeni siten, että graafinen käyttöliittymä oli nopeampi pienemmän hakutulosjoukon käsittelytehtävässä ja tekstipohjainen suuremman hakutulosjoukon käsittelytehtävässä.

Matthias Rauterberg (1992) raportoi graafiselle käyttöliittymälle tehtävästä riippuen yli puolet nopeampia suoritusaikoja tekstipohjaiseen käyttöliittymään verrattuna. Staggers ja Kobus (2000,

174) viittaavat pohdintaosiossaan seitsemään tutkimukseen, joissa graafinen käyttöliittymä on ollut tekstipohjaista käyttöliittymää nopeampi erityisesti navigointitehtävissä. Toisaalta he myös viittaavat kolmeen tutkimukseen, jossa tekstipohjainen käyttöliittymä on ollut graafista käyttöliittymää nopeampi. (Staggers & Kobus, 2000, 174.)

Verkonvalvontaa varten on kehitelty monia erilaisia visualisointeja. Visualisointien hyödynnettävyys ei kuitenkaan ole täysin ongelmatonta. Osa visualisoinneista vaatii tietyn käyttäjäsovelluksen toimiakseen ja osa visualisoinneista on rakennettu vain tiettyä käyttötarkoitusta silmälläpitäen. Visualisointisovellus saattaa olla myös sidottu tietyn valmistajan laitevalikoimaan. (Shiravi, Shiravi & Ghorbani, 2012.)

Myös interaktiivisia visualisointeja on olemassa, mutta suuri osa edellyttää vähintään Java-kielisen sovelluksen suorittamista (ks. esim. Mansman, Meier, & Keim, 2008). Tutkielman pääpaino oli kuitenkin www-selaimen avulla toteutettavan visualisoinnin tutkiminen. Interaktiivisen selainkäyttöliittymän soveltuvuutta verkkolaitteiden porttien tilan visualisointiin ei ole juurikaan tutkittu, jonka vuoksi tässä tutkielmassa esiteltävän visualisoinnin ratkaisut ja kehitystyö perustuivat erilaisilla vapaasti saatavilla olevilla sovelluskehyksillä tehtyihin kokeiluihin.

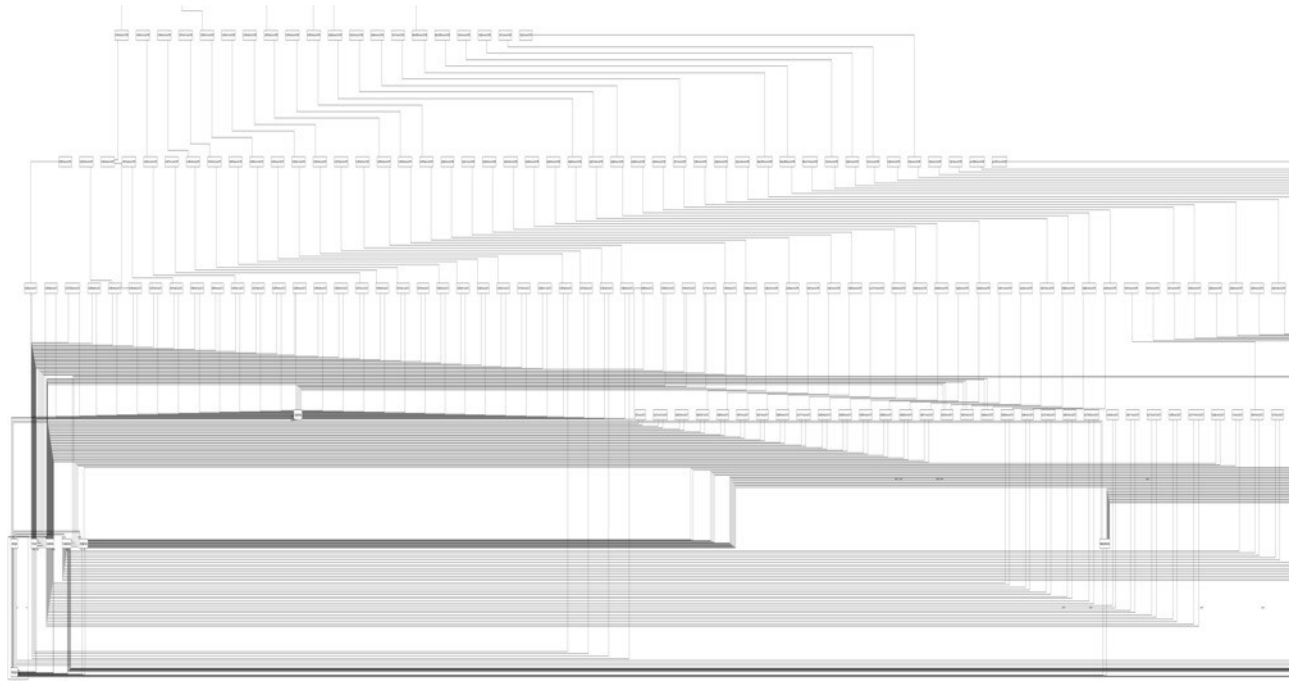
3.3. Asiakas-palvelin -malli ja visualisointikokeiluja

Interaktiivisuus tyypillisesti edellyttää jonkinlaista yhteistoimintaa käyttäjän käskyjen ja niiden reaaliaikaisen prosessoinnin välillä. Eräs tyypillinen toteutustapa on ollut asiakas-palvelin -malli (client-server -model), jossa sovellus on hajautettu ainakin kahdelle eri toimijalle. Esimerkiksi käyttäjän eli asiakasprosessin käskyt, kuten esimerkiksi lähennys, suodatus tai valittujen yksityiskohtien esiintuominen, välitetään tietoverkon avulla erillisellä palvelimella sijaitsevalle palvelinprosessille. Palvelin saattaa edelleen muodostaa tietosisältöjä esimerkiksi tietokantahakujen avulla, joita se tekee tietokantapalvelimelle. Menetelmän hyötynä voi ajatella sen, että asiakaslaitteelta ei edellytetä juurikaan prosessointikapasiteettia ja rivimäärällisesti isojenkin tietokantahakujen prosessointi on nopeaa, jos tietoliikenneyhteydet palvelinten välillä ovat nopeita.

Eräs esimerkki asiakas-palvelin -mallin mukaisesta visualisointisovelluksesta on nimeltään Graphviz. Graphviz on vapaan lähdekoodin sovellus, joka on tehokas, käyttää yksinkertaista komentokieltä ja mahdollistaa monien eri graafityyppien käytön. (Gansner & North, 2000, 1205–1207; ks. myös http://www.graphviz.org/.) Tämän tutkielman kannalta ensimmäinen ajatus olikin kehittää visualisointi hyödyntäen Graphviz-sovellusta: työpöytäkäyttöön tarkoitetulla työasemalla (prosessorina Intel i7-2600, 3,6 GHz) oli mahdollista tuottaa noin 600 verkkolaitteen muodostaman verkon topologiakuva alle sekunnissa. Kuva oli kuitenkin varsin kookas – luettavalla kirjasinkoolla (kahdeksan pistettä) sen koko kasvoi yli 20 000 pisteeseen. Käytännössä kuvasta oli mahdollista tarkastella vain pientä osaa kerrallaan (ks. kuva 4).

Gansner, Hu ja Kobourov (2010) esittelevät menetelmän visualisoida isoja graafeja karttoina. Menetelmä toimii siten, että Graphviz-sovelluksella tuotettuihin karttoihin lasketaan useampi

yksityiskohtien tarkkuustaso, ja nämä tasot toimivat kartan eri kerroksina. Samankaltaisia, toisiinsa yhteydessä olevia, solmuja ryhmitellään tarkennustasosta riippuen yhdeksi tai useammaksi saarekkeeksi, jolloin yleiskuvan saaminen ylätasolla helpottuu. Lopulta käyttäjä saa lisätietoa osoittamalla yksittäistä solmua. Etenkin mobiililaitteita ajatellen menetelmä vaikuttaisi hyvältä lähestymistavalta, sillä koko kartan voisi esiprosessoida valmiiksi. (Gansner et al., 2010; ks. myös http://www2.research.att.com/~yifanhu/MusicMap/index.html.)



Kuva 4. Osa Graphviz-sovelluksen tuottamasta yli 20 000 pistettä leveästä verkkotopologiakuvasta.

Gansnerin ym. (2010) menetelmän esimerkissä käyttöliittymä koostuu esiprosessoiduista kuvista. Menetelmä muutenkin tuntuisi soveltuvan parhaiten ennalta tuntemattomien yhteyksien hahmottamiseen – tietoverkon näkökulmasta menetelmällä voisi toteuttaa esimerkiksi yhteyksien visualisoinnin (ks. alakohta 2.3.3). Tietoverkon topologiakuvan kannalta menetelmä ei ehkä kuitenkaan ole paras mahdollinen, sillä topologia on ennalta tiedossa ja toisaalta esiprosessoidut kuvat eivät kovin hyvin mahdollista esimerkiksi tiedon suodatusta tai yksityiskohtien esilletuomista.

Toisaalta työasemien prosessointiteho on kasvanut ja sovelluksia on mahdollista ohjelmoida alustariippumattomilla ohjelmointikielillä. Laitteistoriippumaton ohjelmointikieli ei tietenkään voi sisältää tietyn prosessoriarkkitehtuurin ominaispiirteitä hyödyntävää konekielistä koodia, joten suoritusnopeus riippuu siitä, miten hyvin sovelluskoodi voidaan tulkata suoritettavaksi. Vaikka tulkattu koodi ei koskaan ole yhtä nopeaa, kuin varta vasten konekielelle käännetty koodi, lisääntynyt prosessoriteho häivyttää tätä eroa. Interaktiivista käyttöä ajatellen asiakkaan käyttöliittymässä tehty tiedon prosessointi on siis hyvä vaihtoehto.

3.4. Selain käyttöliittymänä

Yhä suurempi osa sovellustarjonnasta on käytettävissä tavallisella www-selaimella internetin kautta ”pilvipalveluna”. Esimerkiksi rajoitetulla akkukestolla olevat mobiililaitteet hyötyvät mahdollisimman tehokkaasti suoritetusta koodista. Esimerkiksi Google Chrome -selaimen V8-moottori (https://developers.google.com/v8/) pyrkii kääntämään (tietyillä prosessoriarkkitehtuureilla) JavaScript-kielisen ohjelmakoodin suoraan konekieliseksi koodiksi, jolloin koodin suoritus nopeutuu ja itse sovellus voi olla aiempaa monimutkaisempi.

Myös selainten ominaisuudet ovat kehittyneet. World Wide Web Consortium (W3C, http://www.w3.org) on kehittänyt esimerkiksi www-sivujen toimintaan liittyviä teknisiä suosituksia. Suosituksissa määritellään esimerkiksi se, millaisia elementtejä www-sivulla voi olla ja millaisia ominaisuuksia niiden tulee toteuttaa. Esimerkkejä suosituksista ovat skaalattavat vektorikuvat (scalable vector graphics, SVG) tai cascading style sheet (CSS) -tyylimäärittelyt. W3C:n suositusta voisi ajatella eräänlaisena laadunvarmistuksena: suositusten mukaisesti muotoillun sivun tulisi näyttää samanlaiselta kaikissa suositusta noudattavissa selaimissa. Valitettavasti tämä harvemmin toteutuu käytännössä, sillä tarve standardien noudattamiseen vaihtelee.

Selaimessa tehtävän prosessoinnin eräs rajoitus on se, että JavaScript-koodin suoritusympäristö koostuu vain yhdessä prosessissa. Tällöin muutoin rinnakkaistuva prosessointi ei voi hyödyntää useampaa suoritinydintä (Zakas, 2010, 9). Selaimen tekninen toteutus voi kuitenkin hyödyntää esimerkiksi näytönohjaimen grafiikkaprosessoria (Graphics Processing Unit, GPU) erilaisissa sivun hahmonnukseen (rendering) liittyvissä operaatioissa. Tällöin osa grafiikkaoperaatioista voidaan jättää grafiikkaprosessorin huoleksi ja keskusyksikön tehoa voidaan hyödyntää esimerkiksi tietorakenteen operointiin. Tämä selaimen ja eri tyyppisten suoritinten yhteistoiminta mahdollistaa esimerkiksi grafiikkakirjastoja, joiden avulla käyttäjän selaimessa voidaan visualisoida monimutkaisiakin tietorakenteita kohtuullisella kuormituksella.

3.5. Seurantajärjestelmän suunnitteluperusteita

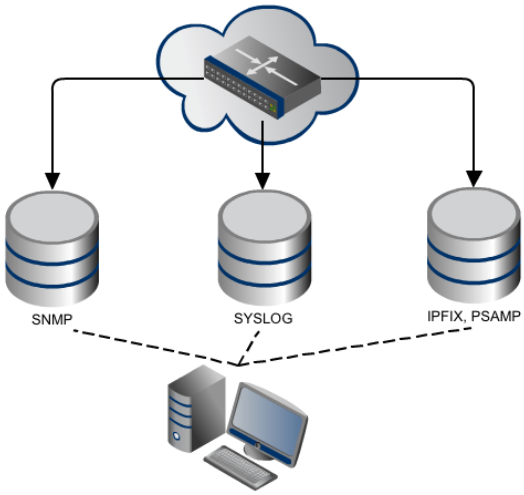
Seurantajärjestelmää suunniteltaessa eräs vaikuttava tekijä on se, miten isoa järjestelmää niillä on tarkoitus seurata ja miten yhtenäinen järjestelmä on esimerkiksi laitekantansa osalta. Tästä syystä isoihin kokonaisuuksiin, kuten esimerkiksi suuryrityksien tarpeisiin, laaditut järjestelmät sisältävät väistämättä erilaisia reunaehtoja siitä, millainen seurattavan järjestelmän tulee olla. Voidaan esimerkiksi ajatella, että ominaisuuksia ei voida täysipainoisesti hyödyntää, jos seurattavat laitteet eivät toteuta kaikkia seurantajärjestelmän toimintoja. Toisaalta seurantajärjestelmät saattavat painottua tiettyjen laitevalmistajien tuotteisiin ja laitevalmistajat ovat kehitelleet omia seurantatuotteitaan, jotka luonnollisesti toimivat parhaiten juuri heidän laitteittensa kanssa.

Voi myös olla, että valmistajat ovat kehitelleet laitteistoihinsa omia menetelmiä havaintotiedon keruulle. Toisinaan tällainen aktiivisuus on hyvästä, sillä esimerkiksi IPFIX:n (ks. alakohta 2.3.3) perustalle on otettu Cisco Systemsin kehittämä ja vapaaseen käyttöön julkaisema Cisco

NetFlow -protokolla (Claise & Wolter, 2007, 201–202). Kaikki valmistajakohtainen kehitystyö ei kuitenkaan saa standardin asemaa, vaan jää taka-alalle. Tällöin tiettyjä keruumenetelmiä hyödyntävä seurantajärjestelmä saattaa olla sidoksissa ja toimia kunnolla vain tietyn valmistajien laitteiden kanssa. Tämä on toki valmistajille edullinen tilanne, mutta pitemmän päälle laitehankinnoissa valinnan mahdollisuutta kaipaavalle kiusallinen, puhumattakaan tilanteista, joissa laitevalmistaja sulautuu toiseen, tai sen toiminta syystä tai toisesta lakkaa.

Työskentely laajan kokonaisuuden kanssa pakottaa ohjelmiston käyttäjän tiettyyn kurinalaisuuteen ja loogiseen jaotteluun ohjelman sisällä. Tämä voi näkyä esimerkiksi seurantaohjelmiston lisäkomponenttien kirjona, syvinä valikkorakenteina tai tarpeettoman raskaana hierarkiana. Käytön kankeus voisi siksi johtaa turhautumiseen ja haluttomuuteen hyödyntää ohjelmistoa täysipainoisesti.

Yleensä kukin seurantatiedon keruujärjestelmä sisältää havaintotietoa vain yksittäisestä havaintotiedon lajista. Tällöin tiedon yhdistely on haastavampaa, sillä kokonaisuuden eri osat ovat hajautuneet useaan eri paikkaan (ks. kuva 5). Niinpä tiedon tarvetta palvelee mahdollisimman monesta järjestelmästä saatavan tiedon keskittäminen yhteen paikkaan, josta voidaan hakea kulloinkin haluttua tietoa.



Kuva 5. Kokonaisvaltainen seuranta pyrkii hyödyntämään kaikkia havaintotiedon lajeja.

Keskittämisellä tähdätään myös siihen, että eri lähteiden havaintotietoa voidaan yhdistellä, jolloin niiden välisiä syy-seuraussuhteita voidaan helpommin selvitellä. Ikävä kyllä Shiravi, Shiravi ja Ghorbani (2012) toteavat, että monet havaintotiedon visualisointiin tarkoitetut järjestelmät on suunniteltu visualisoimaan yksittäistä havaintotiedon tyyppiä, eli esimerkiksi vain SNMP:llä (ks. alakohta 2.3.1) saatua tietoa. Myös Liao, Blaich, VanBruggen ja Striegel (2010) toteavat, etteivät tietoliikenneverkon päivittäisseurantaan luodut yksittäiset työkalut juurikaan edistä kokonaisuuden hahmottamista.

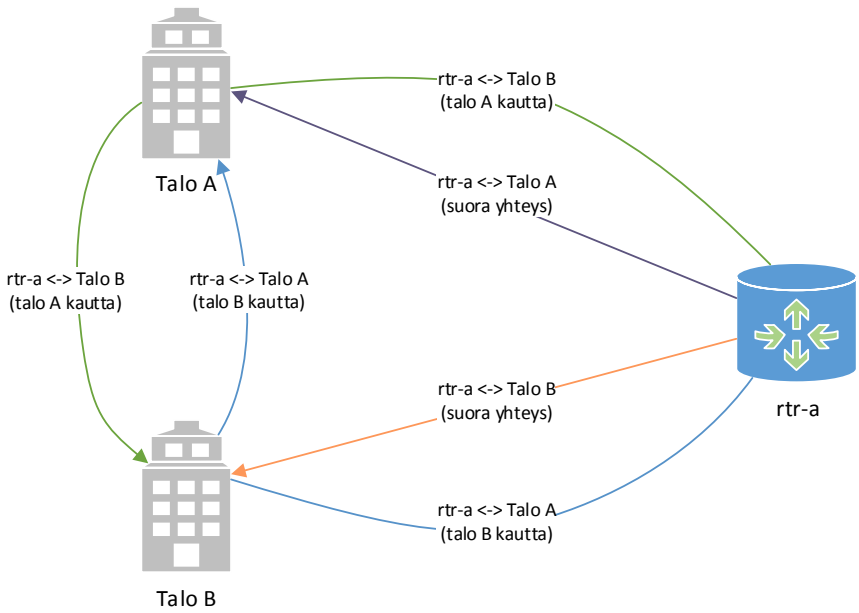
Tämän luvun viimeisessä kohdassa tutustutaan erääseen esimerkkijärjestelmään, jonka suunnittelutavoite on ollut toteuttaa kuvan 5 tietolähteitä kokonaisvaltaisesti hyödyntävä

järjestelmä. Ennen seurantajärjestelmien esittelyä on kuitenkin välttämätöntä tutustua vielä eräisiin verkkotopologian visualisointiin liittyviin suunnitteluvalintoihin.

3.6. Tietoverkon fyysinen ja looginen topologia

Tietoverkossa liikennöivät laitteet voivat viestiä toisten kanssa vain, jos ne on liitetty toisiinsa ja niiden välille voi muodostua tietoliikenneyhteys. Eräs yksinkertaisimmista liitoksista voidaan toteuttaa fyysisenä kytkentänä kahden laitteen välillä. Teknisesti fyysinen kytkentä voidaan toteuttaa esimerkiksi kierretyllä parikaapelilla, valokuidulla tai muilla vastaavilla tekniikoilla. Fyysisellä topologialla tarkoitetaan sitä, miten laitteet ovat fyysisesti kytketty toisiinsa.

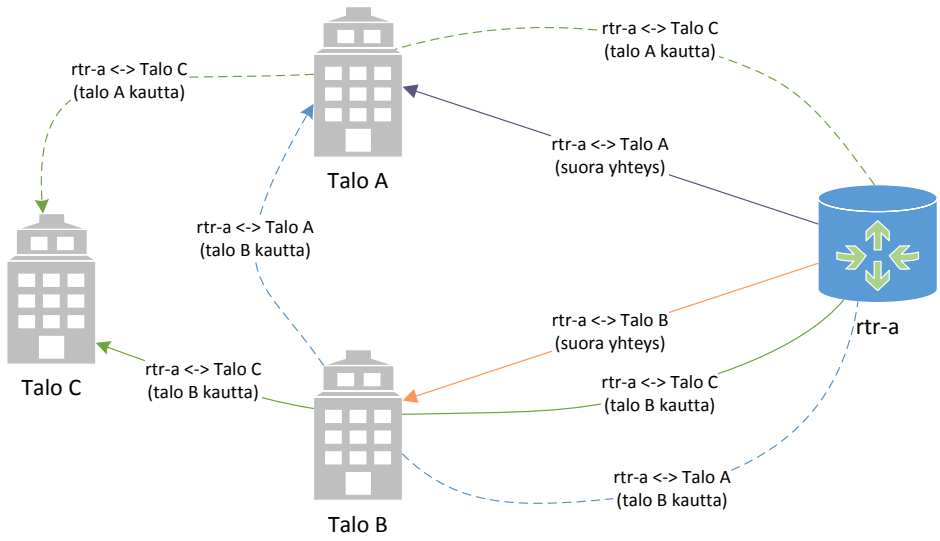
Kuvassa 6 on esimerkki yksinkertaisesta fyysisestä topologiasta. Kuvassa verkkolaite (rtr-a) on yhdistetty kahdella fyysisellä yhteydellä taloihin A ja B. Esimerkiksi yhteys taloon A kulkee siten, että toinen yhteys kulkee suoraan taloon A ja toinen kiertää talon B kautta. Tällä tavoin saavutetaan tilanne, jossa verkkolaitteiden välinen yhteys ei katkea, vaikka jompikumpi fyysisistä yhteyksistä rikkoutuisi esimerkiksi rakennustöiden yhteydessä.



Kuva 6. Fyysinen topologia.

Looginen topologia puolestaan kuvaa sitä, miten laitteiden väliset viestit kulkevat fyysisissä topologioissa. Kuvassa 7 on eräs esimerkki muodostuneesta loogisesta topologiasta. Esimerkiksi verkkolaitteen (rtr-a) ja talon C välisen looginen yhteys ei edellytä suoraa fyysistä yhteyttä, vaan yhteys voi kulkea muiden verkkolaitteiden kautta. Loogisesta yhteydestä ei siis voida suoraan mitata esimerkiksi yhteysnopeutta, sillä nopeus ei riipu sinällään yhden fyysisen yhteyden nopeudesta. Yleisesti ottaen laitteiden välisessä loogisessa yhteydessä voi olla käytössä vain yksi reitti kerrallaan, jolloin muut mahdolliset reitit ovat pois käytöstä (kuvassa 7 katkoviivoin merkityt reitit).

Laitteiden välinen looginen topologia muodostetaan usein jollain yhteyskäytännöllä, kuten esimerkiksi Spanning tree -protokollan (STP) avulla (ks. esim. Perlman, 1985). Tietorakenteena Spanning tree -protokollan lopputulos on puumainen graafi, jonka perusteella jokaiselle verkon aktiivilaitteelle on selvää, miten liikenne verkossa kulkee. STP huolehtii muun muassa siitä, että muodostuneessa STP-puussa laitteiden välisiä fyysisiä yhteyksiä voi olla käytössä vain yksi kappale kerrallaan, jolloin myös verkon looginen rakenne säilyy muuttumattomana.

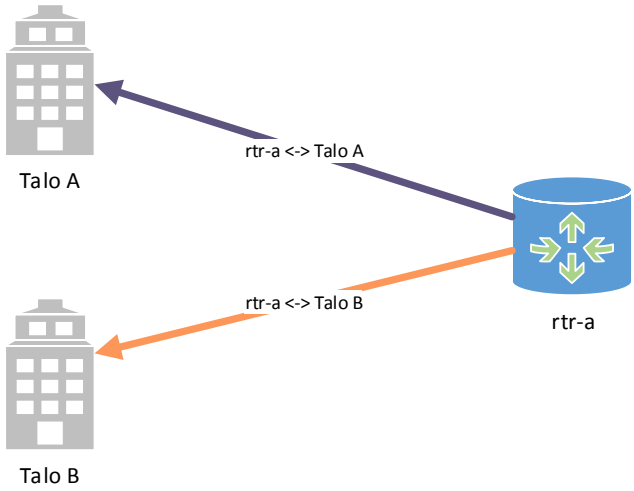


Kuva 7. Eräs esimerkki muodostuneesta loogisesta topologiasta. Katkoviivat osoittavat muita mahdollisia reittejä.

Eräs erikoistapaus fyysisestä topologiasta on verkkolaitteiden välisten yhteyksien niputtaminen (link aggregation tai link bundling) yhdeksi. Tällöin siis esimerkiksi niputettaessa kaksi yhden gigabitin sekuntinopeudella toimivaa yhteyttä saadaan yksi kahden gigabitin sekuntinopeudella toimiva yhteys (ks. kuva 8). Niputuksen etu kahteen yksittäiseen yhteyteen verrattuna on se, että tietoverkko, jossa käytetään esimerkiksi edellä mainittua Spanning tree -protokollaa, voi niputtamattomissa yhteyksissä hyödyntää vain toista fyysistä yhteyttä kerrallaan. Niinpä jos pääasiallisesti käytetty yhteys katkeaa, toinen yhteys kyllä otetaan käyttöön. Varayhteyden käyttöönoton yhteydessä liikenne kuitenkin katkeaa niin pitkäksi aikaa, kunnes topologiapuu on muodostettu uudelleen. Sen sijaan niputetuissa yhteyksissä verkkolaitteet tasaavat verkkoliikenteen molemmille yhteyksille ja toisen yhteyden katkeaminen näkyy verkon käyttäjälle vain yhteysnopeuden laskuna, sillä STP-topologiapuuta ei tarvitse muodostaa uudelleen.

Topologian käsitteiden avulla voidaan paremmin ymmärtää ratkaisuja, joita visualisointia kehitettäessä väistämättä täytyy tehdä. Visualisointia koostettaessa täytyy esimerkiksi valita, mallinnetaanko fyysisistä vai loogista topologiaa. Jos esimerkiksi hahmoteltaisiin visualisointia verkon käyttöasteesta: yksittäisen fyysisen kytkennän muodostaman yhteyden mittaus antaa tietoa juuri kyseisestä yhteydestä, mutta toisaalta niputetun yhteyden visualisointi kertoo kokonaiskuvan yhteysvälin toiminnasta ja kulloisestakin nopeudesta. Monimutkainen verkkotopologia voi sisältää

useita yksittäisiä fyysisiä yhteyksiä, joista suuri osa on niputettu yhdeksi yhteydeksi ja tällöin täytyy valita se, visualisoidaanko niputetun yhteyden kokonaiskuorma, vai siihen liittyvien fyysisten yhteyksien kuorma yhteys kerrallaan.



Kuva 8. Fyysinen topologia (niputettu yhteys).

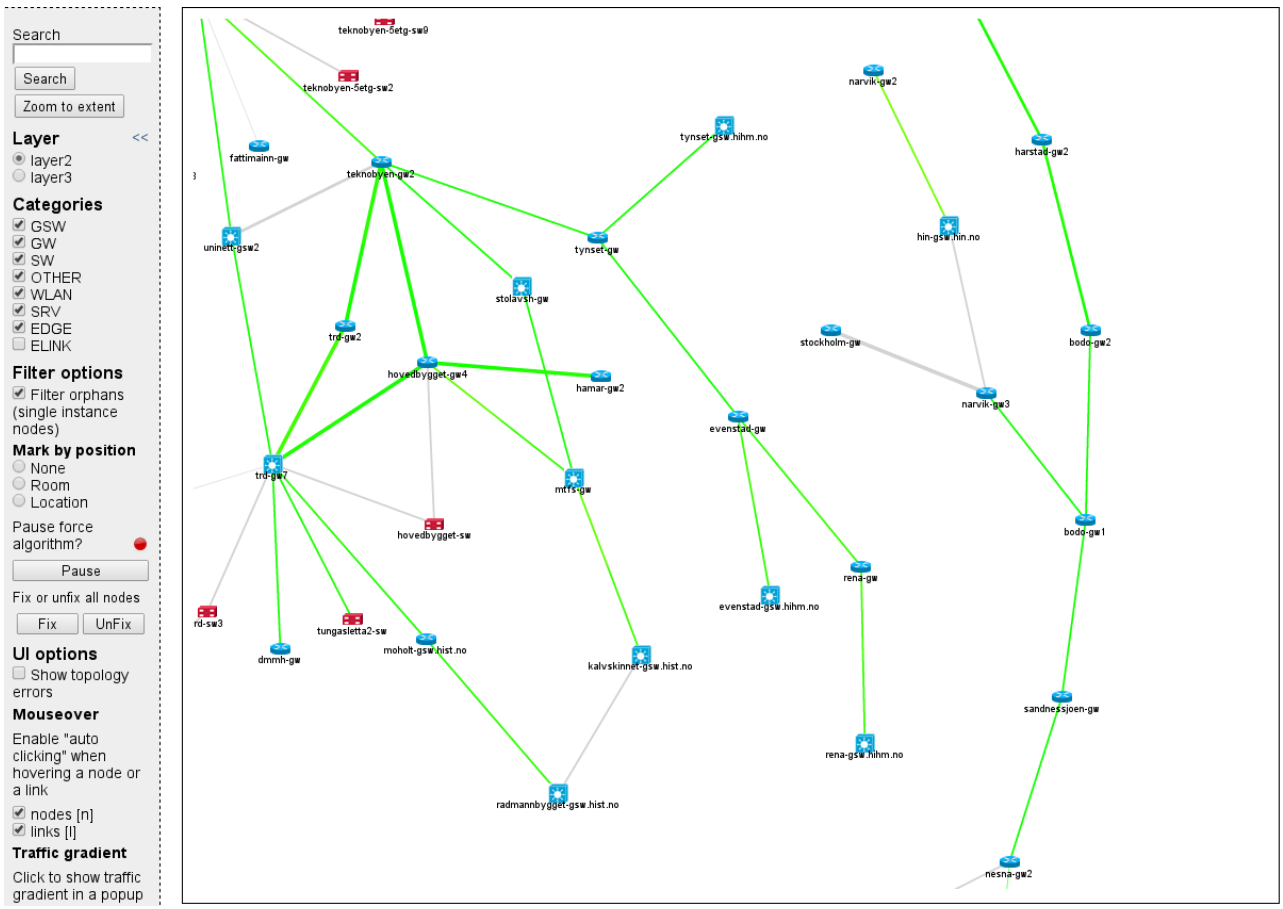
Loogisen topologian visualisoinnissa tulee varautua siihen, että visualisointi voi muuttua riippuen siitä, millainen on laitteiden muodostama topologiapuu. Toisaalta esimerkiksi yhteyksien näytteistystä (ks. alakohta 2.3.3) voitaisiin hyödyntää poikkeavan verkkoliikenteen visualisointiin. Esimerkiksi Liao ja muut (2010) ovat luoneet sovelluksen nimeltä ENAVis (Enterprise Network Activities Visualization), jonka tavoitteena on havainnollistaa yhteyksiä palvelimen, käyttäjän ja sovelluksen välillä niin, että palvelinten välisen tietoverkon valvontatietoa hyödyntämällä saadaan mukaan myös palvelinprosessien välinen viestintä. Sovellus hyödyntää IPFIX:in pohjalla olevaa NetFlow-protokollaa, SNMP:tä ja palvelimilla sekä työasemilla ajettavia valvontaprosesseja. Hyvänä suunnitteluperusteena on siis valita visualisoinnin toteutus siten, että samalla pohjatyöllä on mahdollista visualisoida mahdollisimman montaa havaintotiedon lajia.

Topologian ohella visualisoinnin toinen suunnitteluratkaisu liittyy siihen, miten visualisoinnin tietorakenne toteutetaan. Useat eri fyysiset yhteydet laitteiden välillä aiheuttavat sen, että graafista tulee syklinen. Syklisyydellä tarkoitetaan sitä, että solmuun voi päästä takaisin toista reittiä ja tällöin ei välttämättä ole selvää, kumpi solmuista oli jälkeläissolmu. Selvitys on toki mahdollista erilaisten graafialgoritmien avulla joka tavallaan muistuttaa Spanning tree -protokollan käyttöä, mutta monimutkaistaa tietorakenteen muodostamista.

Tämän tutkielman myöhemmissä luvuissa pidättäydytään yksinkertaisissa tietorakenteissa, joista muodostetut graafit ovat asyklisiä. Tietorakenteet ovat siten myös valmiiksi hierarkkisia ja solmujen välillä on vain yksi yhteys. Tietorakenteet siis mallintavat pitkälti kuvan 8 mukaista fyysistä topologiaa.

3.7. Erilaisia seurantajärjestelmiä

Tämän tutkielman johdannossa esiteltiin yksinkertainen ”sääkartta” sekä yhteysvälin kuormitusta havainnollistava kuvaaja (ks. kuvat 1 ja 2). Kuvissa ei kuitenkaan ole interaktiivisia ominaisuuksia ja tietosisältö rajoittunee käytännössä SNMP:n (ks. alakohta 2.3.1) avulla haetun porttitiedon visualisointiin. Astetta monimutkaisempi esimerkki tietoverkon seurantajärjestelmästä on Norjan opetuksen ja tutkimuksen tietoverkon (UNINETT, https://www.uninett.no) kehittämä NAV-sovellus (Network Administration Visualized, https://nav.uninett.no/). NAV:in idea on tarjota ylläpitäjälle työkalupakki, jonka avulla havaintotietoon voidaan kohdistaa hakuja tai esimerkiksi verkkotopologian visualisointeja. Eräs visualisointi on verkon topologiakuva, joka muodostetaan D3-grafiikkakirjastolla käyttäen voimavektoreihin perustuvaa visualisointimallia (Force-directed graph layout, ks. https://github.com/mbostock/d3/wiki/Force-Layout) topologian visualisointiin (ks. kuva 9).



Kuva 9. Esimerkki verkon topologiakuvasta NAV-järjestelmän versiossa 3.3. (lähde: https://nav.uninett.no/).

Kuvassa 9 esitelty topologian visualisointi rajoittuu runkoverkon tasolle. Johdannossa ideoidun lähituen työntekijän tarpeita varten visualisointiin tarvittaisiin mahdollisuus porautua syvemmälle topologiaan. Toisaalta kaikkien asiakaskytkentäisten verkkolaitteiden porttitason visualisointi

samalla kertaa tekisi visualisoinnista todennäköisesti sekavahkon ja raskaan, joten jonkinlainen interaktiivinen visualisointi on eittämättä tarpeen.

Tätä silmälläpitäen tutkielman alkuvaiheessa luotiin visualisointi, jossa kytkinportteja kuvaavat alisolmut saadaan näkyviin ja pois näkyvistä hiiren osoituksen avulla. Tutkielman seuraavissa luvuissa pyritään mittaamaan sitä, miten nyt kehitetyn interaktiivisen topologiavisualisoinnin avulla voidaan hakea tietty tieto ja miten käytettävä tällainen käyttöliittymä on. Mittaus vertailee luvussa 2 esiteltyjen automaattisten menetelmien avulla kerättyjen havaintotietojen pohjalta luotua visualisointia suoraan tekstipohjaiseen kommunikaatioon. Mittaus on tärkeää toteuttaa, sillä tulosten perusteella voidaan arvioida kuvan 9 kanssa samaan menetelmän perustuvan graafisen käyttöliittymän soveltuvuutta verkkoylläpidon tai lähituen päivittäiseen käyttöön.

4. Menetelmä

Tässä luvussa esitellään tutkielmassa käytetyn visualisoinnin toteutus, osallistujat, mittauksessa käytettävä laitteisto, mittausmenetelmät käyttäjätyytyväisyydelle sekä käyttöliittymille, tehtävät sekä tutkimuksen kulku, ja mittauksessa saadun aineiston analyysi.

4.1. Visualisoinnin toteutus

Visualisoinnin käyttöliittymäksi oli luontevaa valita www-selain ja varsinaisen toteutuksen pohjaksi jokin avoimen lähdekoodin sovelluskehys (framework). Sovelluskehystä valitessa tutkittiin erilaisia vaihtoehtoja, kuten Arbor.js (http://arborjs.org/), Data Driven Documents (D3) (http://d3js.org/), Raphaël (http://raphaeljs.com/) ja aiemmin mainitun Graphviz:n Webdot -palvelu (http://api.graphviz.org/webdot/index.html). Näistä Arbor.js, D3 ja Raphaël ovat ikään kuin puhdasverisiä JavaScript-kehyksiä, joissa prosessointi tapahtuu selaimessa. Webdot on puolestaan rajapinta, jonka avulla graafin piirto voidaan suorittaa www-palvelimella hyödyntäen Graphviz-sovellusta (ks. kohta 3.3).

Sovelluskehykseksi valittiin Data Driven Documents (D3). D3 on Mike Bostockin, Jeff Heerin ja Vadim Ogievetskyn (2011) kehittämä JavaScript-kielinen visualisointikirjasto. D3 tuottaa selaimessa W3C:n (ks. kohta 3.4) suosituksia noudattavia visualisointeja, eli käytännössä skaalattavia vektorikuvia. Toisin sanoen D3-visualisointikirjaston pitäisi toimia jokaisella selaimella, joka noudattaa W3C:n standardeja ja jossa on mahdollisuus JavaScript-koodin suorittamiseen. Käyttöliittymä toimii osittain sellaisenaan esimerkiksi mobiililaitteiden kosketusnäytöillä. Mobiilikäytön erityispiirteiden tarkastelu kuitenkin rajattiin tämän tutkielman ulkopuolelle.

D3:n valintaa puolsi laaja selainyhteensopivuus, valmis tuki interaktiivisuudelle ja aktiivinen kehitystyö. Lisäksi D3:n arkkitehtuuri mahdollistaa visualisointimantran (ks. kohta 3.1) toteuttamisen yksinkertaisella tavalla. On kuitenkin huomattavaa, ettei D3 välttämättä edusta millään tavoin poikkeuksellista edistyksellisyyttä vaan samat ominaisuudet ovat toteutettavissa myös muihin grafiikkakirjastoihin. Toisaalta esimerkiksi kohdassa 3.3 esitelty Graphviz:in Webdot-rajapinta sisältää monipuolisempia visualisointialgoritmeja, mutta hintana on heikompi interaktiivisuus, sillä graafit ovat aina valmiita kuvia ilman interaktiota. Eri visualisointikirjastoista D3 siis vastasi parhaiten tämän tutkielman tarpeita.

Visualisoinnissa käytetään D3-kirjaston voimavektoreihin perustuvaa visualisointimallia. Malli perustuu Barnes-Hutin approksimaatioon, jolla voidaan simuloida kappaleiden keskinäisiä voimasuhteita. Samankaltainen malli on käytössä myös muilla (ks. esim. Mansman, Meier, & Keim, 2008). Approksimaation lisäksi voimavektorimallissa voidaan parametroida esimerkiksi solmujen välisen yhteyden toivottu pituus, joka käytännössä rajoittaa solmujen keskinäistä etäisyyttä. Mallin teoreettinen ymmärrys tai yksityiskohtainen teknisen toteutuksen kuvaus ei ole

olennaista tämän tutkielman kannalta, joten seuraavaksi käydään läpi menetelmän toiminta yleisellä tasolla.

Voimavektorimallissa kullekin solmulle on asetettu oletusvaraus. Tällöin samanmerkkiset varaukset hylkivät toisiaan. Toisaalta hylkimistä rajoittaa esimerkiksi parametroitavat vakiot, kuten aiemmin mainittu linkin pituus. Simulaatio pysähtyy, kun solmun etäisyyssiirtymä alittaa tietyn parametroitavan raja-arvon. Raja-arvoa muuttamalla voidaan kontrolloida laskentatehon kulutusta. Toisaalta simulaation pysäytyksellä estetään se, että solmujen pieni keskinäinen, värinänä näkyvä, liike ei häiritse käyttäjää. Simulaation lopputuloksena on visualisointi, jonka solmut ovat sijoittuneet optimaalisesti toisiinsa nähden. Visualisointi on siten mahdollisimman tiivis esitys tietorakenteesta.

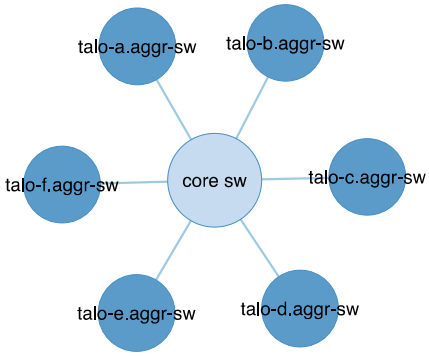
Toinen merkittävä ominaisuus D3:ssa on se, että sen arkkitehtuuri mahdollistaa helposti toteutettavan interaktiivisen tietorakenteen visualisoinnin. Kehitetyn visualisoinnin tietorakennetta voidaan avata ja sulkea esimerkiksi osoittamalla solmua hiirellä. Uusi simulaatio käynnistyy ja mikäli solmulla on jälkeläissolmuja, ne näytetään tai poistetaan näkyvistä. Simulaation yhteydessä solmut piirretään uudelleen ruudulle. Tällöin olisikin mahdollista päivittää lähes reaaliaikaisesti taustalla olevan tietorakenteen yksityiskohtia.

D3 tukee useita menetelmiä tietorakenteen lataamiseksi. Tässä tutkielmassa toteutetun visualisoinnin tietorakenne ladataan selvyyden vuoksi palvelimelta yhtenä JSON-objektina (RFC 7159), mutta sinällään mikään ei olisi estänyt myös ositetun rakenteen käyttöä. Tietorakenne koostui objekteista (solmut) ja niiden aliobjekteista (jälkeläissolmut). Objektit sisältävät niille ominaisia attribuutteja, kuten solmun tyypin, nimen, kuvauksen tai tilatiedon. Attribuutit ovat yksinkertaisimmillaan avain-arvopareja, jolloin niitä on helppo lisätä tai poistaa tietorakenteesta. Tällöin toteutetussa visualisoinnissa voidaan huomioida solmun erityispiirteet ja esittää esimerkiksi haluttuja lisätietoja.

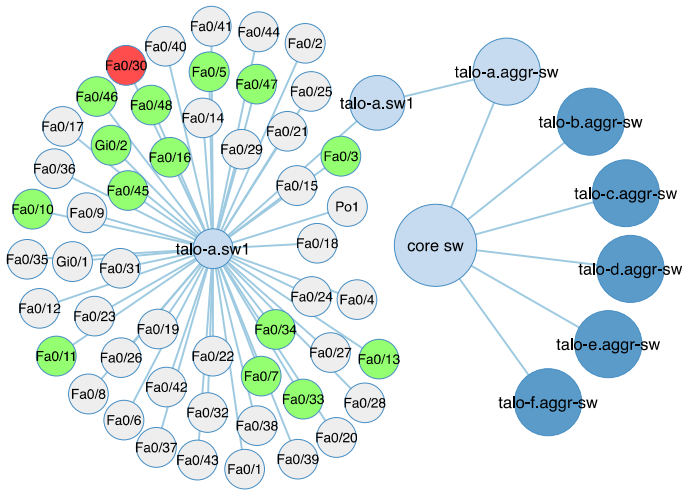
Tässä tutkielmassa toteutetun visualisoinnin käynnistysvaiheessa tietorakenteesta vain ensimmäinen kerros on avattuna (ks. kuva 10). Visualisoinnissa hyödynnettiin tietorakenteeseen tallennettua tietoa solmutyypistä. Käyttöliittymässä solmua edustaa täytetty ympyrä ja solmujen välinen linkki on viiva. Ympyrän (solmun) täyttöväri vaihtui tilan mukaan: täyttöväri oli tummansinen silloin, kun solmulla on jälkeläissolmuja, mutta ne eivät olleet näkyvissä. Täyttöväri muuttui vaaleansiniseksi silloin, kun solmu oli avattu eli sen jälkeläissolmut ovat näkyvissä. Vaihtoehtona värisymboleille oli käyttää NAV:n tapaan ikoneja (ks. kuva 9), mutta tästä luovuttiin, sillä värien käyttö oli teknisesti yksinkertaisempaa toteuttaa.

Varsinainen värien suunnittelu keskittyi kytkinportti-tyyppisiin solmuihin. Värit määriteltiin vaihtelevan harmaan (ei kytketty), vihreän (kytketty) ja punaisen (virhe) välillä (ks. kuva 11). Solmujen värit valittiin olemaan riittävän etäällä toisistaan ja toisaalta noudattamaan yleisiä värikoodauksen konventioita, kuten punainen vaaran tai vihreä elämän merkkinä (Ware, 2013, 126–127). Puna-vihervärisokeus huomioitiin mittauksen kyselylomakkeessa erikseen. Kytkinportti-tyyppisissä solmuissa käytettiin myös pienempää kirjasinkokoa, koska tyypillisesti verkkolaitteiden

porteista on saatavilla jonkinlainen lyhennetty nimi (esimerkiksi Cisco Systemsin laitteissa Fa0/1 tai Eth3/25). Tätä kautta myöskään visualisoinnin koko ruudulla ei kasva tarpeettomasti.



Kuva 10. Sovelluksen oletusnäkymä.

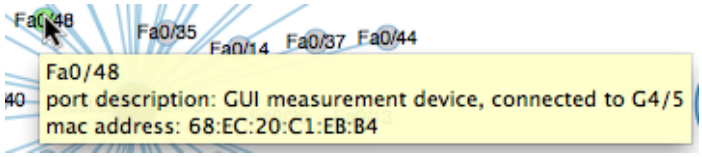


Kuva 11. Solmu ja kytkinportit visualisoituna.

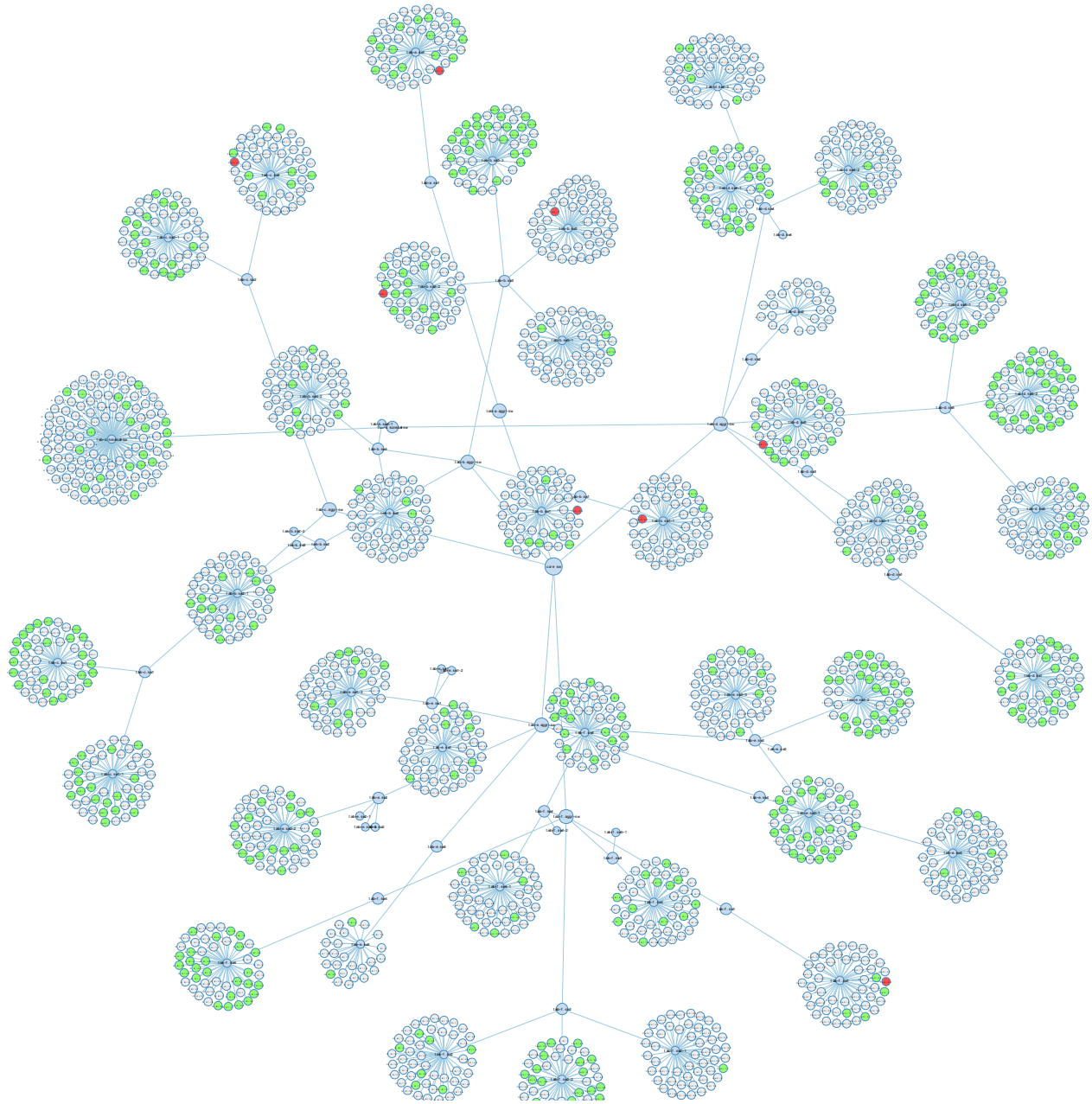
Yksittäisille solmuille toteutettiin myös ominaisuus, jossa solmun päälle viety hiiren osoitin tuo esille vihjelaatikon (tooltip). Vihjelaatikon sisältö vaihteli sen mukaan, millaisia attribuutteja solmulla oli. Eräs esimerkki vihjelaatikosta on esitetty kuvassa 12, jossa esille tuodaan portin kuvaus (port description) ja verkkosovittimen osoite (mac address). Toiminnallisuuden tarkoitus oli toteuttaa visualisointimantran viimeinen vaihe, eli tuoda esille yksityiskohdat kysynnän mukaan. Vihjelaatikkoon voisi jatkossa liittää esimerkiksi toiminnallisuuksia, kuten kytkinportin aliverkon vaihdon tai portin kuvauksen muuttamisen. Vihjelaatikkoon voisi sijoittaa myös erilaisin tiedonkeruumenetelmin haettua tietoa, kuten esimerkiksi verkkosovittimen osoitetta vastaava IP-osoite, laitteen verkkonimi tai linkki muihin järjestelmiin. Vastaavat toiminnallisuudet olisi mahdollista sijoittaa myös solmujen välisiin linkkeihin.

Tietorakenne voi olla avattuna useammasta kohdasta, jolloin verkkotopologiasta on mahdollista tarkastella esimerkiksi kuvan 13 mukaista yleiskuvaa. Lähennystä ja loitonnusta varten visualisointiin kehitettiin myös toiminnallisuus, jossa näkymän lähennystä tai loitonnusta voi

kontrolloida esimerkiksi hiiren rullalla. Tässä mielessä D3 ei aseta solmujen lukumäärän suhteen rajoituksia, mutta JavaScript-ajoympäristön arkkitehtuurin vuoksi yksittäisen suoritinytimen laskentateho asettaa käytännön rajat solmujen lukumäärälle.

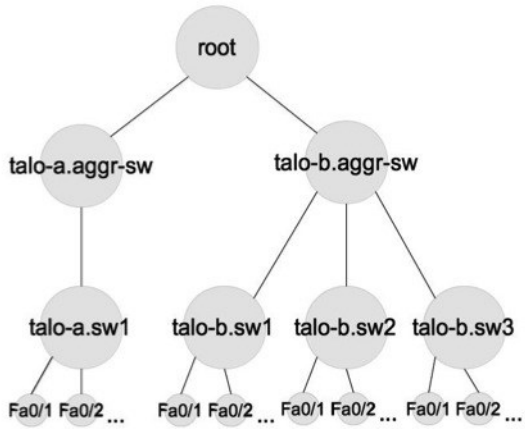


Kuva 12. Vihjelaatikon sisältö.



Kuva 13. Esimerkkivisualisointi loitonnettuna yleiskuvaa varten.

Osoituksiin toteutettiin toiminnallisuus, jossa hiiren osoitus avasi tietorakennetta kahdella tavalla: jos solmulla oli vain yksi jälkeläissolmu, jälkeläissolmuja avattiin niin kauan, kunnes solmuja ei löytynyt tai solmulla oli yhtä useampi jälkeläissolmu. Vastaavasti jos jälkeläissolmuja oli useampia jo osoitusvaiheessa, osoitus toi esille vain seuraavan tason. Tällöin esimerkiksi kuvan 14 mukaisesti osoitettaessa solmua *talo-a.aggr-sw*, käyttöliittymä avaisi myös solmun *talo-a.sw1*. Jos osoitettaisiin solmua *talo-b.aggr-sw*, käyttöliittymä avaisi näkyviin solmut *talo-b.sw1*, *talo-b.sw2* ja *talo-b.sw3*. Toiminnallisuuden tarkoitus oli vähentää tarvittavien osoitusten määrää.



Kuva 14. Esimerkki tietorakenteesta.

Kehitystyön aikana tiedostettu haittapuoli voimavektoreihin perustuvassa menetelmässä on, että menetelmä ei suoraan ota kantaa esimerkiksi solmujen piirtojärjestykseen tai keskinäiseen sijaintiin. Niinpä jälkeläissolmujen asettuminen ei noudata mitään erityistä mallia, eli niiden käytös ei sinänsä ole ennustettavissa. Tämä luonnollisesti heikentää visualisoinnin tehokkuutta erityisesti silloin, kun solmulla on useita kymmeniä jälkeläisiä. Ennustettavuutta olisi mahdollista parantaa esimerkiksi parametroimalla algoritmia tai laskemalla jälkeläissolmujen x- ja y-koordinaatit etukäteen. Tämä kehitystyö kuitenkin jätettiin tulevaisuuteen.

4.2. Osallistujat

Mittaukseen osallistui 10 vapaaehtoista henkilöä, joista miehiä oli yhdeksän kappaletta ja naisia yksi kappale. Osallistujat olivat iältään 29–37 vuotta ja keski-ikä oli 33,6 vuotta. Osallistujat oli jaettu joko eksperttiryhmään (n = 5) tai noviisiryhmään (n = 5) sen perusteella, kuuluiko heidän päivittäiseen työhönsä verkkolaitteiden ylläpito tekstipohjaisen käyttöliittymän avulla vai ei.

Osallistujien käyttökokemusta tekstipohjaisista käyttöliittymistä selvitettiin taustatietolomakkeella (ks. liite 1). Eksperttiryhmällä oli käyttökokemusta tekstipohjaisista käyttöliittymistä keskimäärin 16 vuotta ja käyttö oli ollut jatkuvaa tai vähintään monta kertaa päivässä toistuvaa. Noviisiryhmällä käyttökokemusta tekstipohjaisista käyttöliittymistä oli keskimäärin 14,6 vuotta. 80 prosenttia eksperttikäyttäjistä käytti tekstipohjaista käyttöliittymää jatkuvasti ja 20 prosenttia useita kertoja päivässä. 60 prosenttia noviisikäyttäjistä käytti jonkin

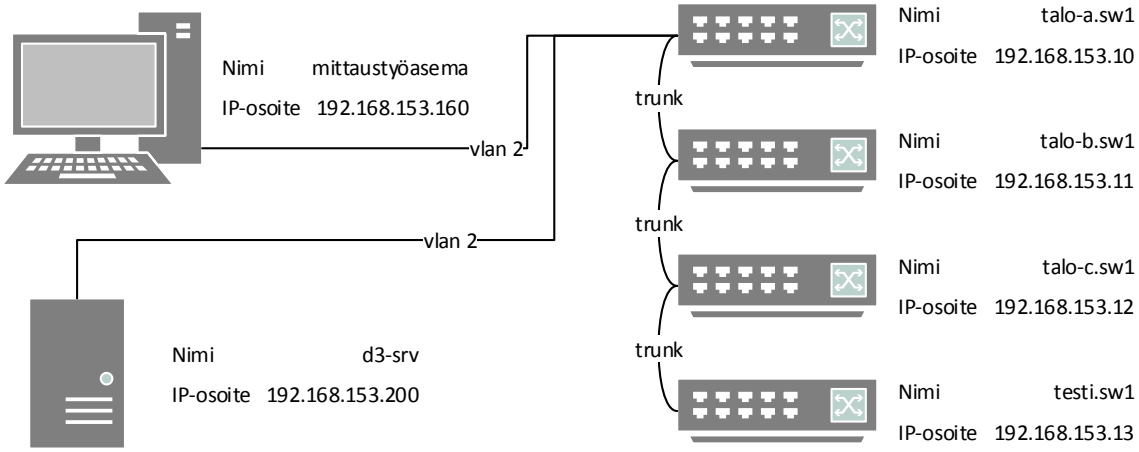
sovelluksen tekstipohjaista käyttöliittymää jatkuvasti. Tyypillisin esimerkki käytöstä oli Internet Relay Chat (IRC) -keskusteluihin osallistuminen tai käyttöjärjestelmän komentokehotteen hyödyntäminen työtehtävien suorittamisessa. 20 prosenttia käytti tekstipohjaista käyttöliittymää 1–2 kertaa viikossa ja 20 prosenttia harvemmin kuin kerran viikossa.

Yhdellä mittaukseen osallistujalla oli todettu puna-vihervärisokeus. Harjoitteluvaiheessa käyttöliittymään valittujen värien havainnointi ei tuottanut ongelmia, joten osallistuja suoritti tehtävät normaalisti.

4.3. Laitteisto ja tekniset asetukset

Mittalaitteiston tietoverkon muodostivat neljä kappaletta Cisco 2960 -kytkimiä, joita käytettiin tekstipohjaisen käyttöliittymän tehtävissä sekä mittausaineiston siirrossa. Kytkinten käyttöjärjestelmänä oli Cisco IOS 15.0. Kytkimistä yksi (*testi.sw1*) oli varattu harjoittelulle ja loput kolme (*talo-{a,c}.sw1*) tehtävien tekoon. Kytkimet oli yhdistetty toisiinsa parikaapeleilla ja porttien asetukset muokattu siten, että tehtävissä esiintyvät tilanteet muodostuivat. Mittauksessa päädyttiin käyttämään oikeita verkkolaitteita. Tämä johtui siitä, että vapaasti saatavilla olevat verkkosimulaattorit (esim. GNS3, http://www.gns3.net) ovat pääasiassa kehitetty simuloimaan (reitittyvää) verkkoliikennettä, jolloin niiden ”laitevalikoima” keskittyy lähinnä verkon runkolaitteisiin. Toisaalta oikeita verkkolaitteita oli helposti saatavilla, jolloin valinta oli yksinkertainen.

Kytkinten hallintaliittymät, eli liittymät, joihin mittaustyöasemalta otettiin yhteyttä, oli sijoitettu samaan virtuaaliseen aliverkkoon (Virtual LAN eli VLAN) mittalaitteiston kanssa (ks. kuva 15). Muita kytkinten portteja ei ollut kytketty tähän aliverkkoon, joten mittaukseen liittyvä informaatio ei missään vaiheessa olisi ollut saatavilla muista kytkinporteista. Toisaalta mittausta ei myöskään olisi päässyt häiritsemään muiden kytkinporttien kautta. Kytkinten asetuksia ei muutettu mittauksen aikana. Sisäverkosta ei ollut yhteyksiä muihin verkkoihin. Nimipalvelukyselyjä varten */etc/hosts*-tiedostoon sijoitettiin IP-osoitteet ja niitä vastaavat verkkonimet kuvan 15 mukaisesti.



Kuva 15. Mittaustietoverkon asetukset.

Varsinaiset mittaukset suoritettiin MacBook Pro -kannettavalla, johon oli kytketty ulkoinen näyttö, näppäimistö ja hiiri. Käyttöjärjestelmänä oli OS X 10.8.5, suorittimena 2,53 GHz Core 2 Duo ja muistia 8 gigatavua. Näyttö oli kooltaan 24” (Samsung 245B plus) ja tarkkuus 1920 x 1200 pistettä. Käyttäjäkohtaiset asetukset nollattiin ennen jokaista mittausta. Kaikki käyttöliittymätestaukset tehtiin Mozilla Firefox -selaimella, jonka versio oli 26.0 (http://www.mozilla.org/firefox/).

Mittalaitteiston tehtävät sekä visualisointisovellus ladattiin erilliseltä mittausverkon palvelimelta. Palvelin oli helpon siirrettävyyden vuoksi virtuaalikone, jonka käyttöjärjestelmänä oli Ubuntu Linux, jonka jakeluversio oli 13.10. Palvelimelle tallennettiin molempien käyttöliittymien tuottama mittausaineisto siten, että graafisen käyttöliittymän tuottama mittausaineisto tallennettiin reaaliaikaisesti (ks. alakohta 4.5.1) ja tekstipohjaisen käyttöliittymän mittausaineisto mittauksen päätyttyä. Palvelin varmuuskopioitiin neljä kertaa vuorokaudessa, jotta mahdollisen laiterikon vaikutukset jäisivät mahdollisimman vähäisiksi.

4.4. Käyttäjätyytyväisyys

Mittauksessa haluttiin selvittää käyttöliittymään liittyviä subjektiivisia tuntemuksia, jota varten laadittiin kyselykaavake. Kaavake muistutti käyttäjätyytyväisyyttä mittaavaa Questionnaire for User Interface Satisfaction (QUIS) –kyselyä (ks. esim. Chin, Diehl & Norman, 1988). QUIS-kyselyn asteikot puolestaan ovat tyypiltään semanttista differentiaalia, joka on perustaltaan kohtuullisen hyvin tutkittu ja suhteellisen vakiintunut tunnetutkimuksen menetelmä (Osgood, 1952; Bradley & Lang, 1994; Surakka, Illi & Isokoski, 2004; Vanhala, 2005). Semanttinen differentiaali itsessään soveltuu siis hyvin käyttäjätyytyväisyyden mittaamiseen (Vanhala, 2005).

Kyselykaavake (ks. liite 4) koostui yhdeksästä kaksinapaisesta asteikosta, joiden ääripäissä olivat keskenään vastakkaiset adjektiivit ja keskiosa ilmensi neutraalia suhtautumista. Kaavakkeessa esitetyt arviointiasteikot olivat: yleinen, miellyttävyys, nopeus, tehokkuus, vaikeus, tiedon esitystapa ja käyttökelpoisuus. Kaavake esitettiin ja selitettiin testiin osallistujalle kummankin käyttöliittymän mittaukseen liittyvien navigointitehtävien päätteeksi.

4.5. Mittausmenetelmät

Tässä kohdassa esitellään käyttöliittymiin liittyvien mittausten tekninen toteutus. Yksittäiseen tehtävään liittyvä mittaus koostuu suoritusajan ja virhemäärän mittauksesta. Varsinaisten instrumentointitoteutusten esittely jakautuu kahteen alakohtaan: alakohdassa 4.5.1 esitellään graafisen käyttöliittymän mittauksien tekninen toteutus ja alakohdassa 4.5.2 vastaavasti tekstipohjaisen käyttöliittymän mittauksien tekninen toteutus.

Kaikki mittaukset suoritettiin kontrolloiduissa laboratorio-olosuhteissa. Osallistujan toimintaa seurattiin käyttöliittymiin rakennettujen instrumentointien avulla sekä mittauksen vetäjän toimesta.

4.5.1. Graafinen käyttöliittymä

Lazar, Feng ja Hochheiser (2010) esittelevät joukon instrumentointeja, joilla selainpohjaisen käyttöliittymän suoritusaikoja voi mitata. Eräs tehtävän suorittamiseen kuluvan ajan mittaus

voidaan toteuttaa yksinkertaisesti www-palvelimen lokien avulla (Lazar, Feng & Hochheiser, 2010). Www-palvelin oli sijoitettu samaan aliverkkoon mittaustyöaseman kanssa, jotta verkon viiveistä aiheutuvat mittausepätarkkuudet minimoituisivat. Toisaalta viiveen pitäisi olla likimain sama kaikille. Www-palvelimen lokiriveistä käyvät ilmi ajankohdat, jolloin mittauksen tehtäväsivu on jaettu käyttäjälle, joten testaus toteutettiin tekemällä alkuun harjoittelusivu. Harjoittelusivulta siirryttiin ensimmäiseen tehtävään ja tehtävät oli linkitetty toisiinsa niin, että tehtävän lopuksi siirryttiin seuraavalle tehtäväsivulle.

Tehtäväkaavake (ks. liite 3) oli koko ajan osallistujan nähtävillä. Tällöin huomioitavana vaarana oli se, että osallistuja vahingossa suorittaa annetut tehtävät kerralla, eikä aloita tehtäviä ikään kuin perustilasta. Niinpä mittauksen lähdeaineisto laadittiin graafisessa käyttöliittymässä siten, että parittomissa tehtävissä kysyttyjä tietoja ei ollut mahdollista hakea parilliseen tehtävään liittyvästä lähdeaineistosta.

Www-palvelinsovelluksena käytettiin Apache HTTPD (http://httpd.apache.org/) versiota 2.4. Apache HTTPD on avoimen lähdekoodin palvelinsovellus, joka on saatavilla yleisimpiin Linux-jakeluihin. Versio 2.4 valittiin sen vuoksi, että tässä versiossa on suoraan mahdollisuus määritellä syntyvään lokimerkintään ajan tarkkuudeksi myös millisekunnit (ks. esim. http://httpd.apache.org/docs/2.4/mod/mod_log_config.html). Esimerkki syntyvästä lokirivistä on alla:

```
192.168.53.131 [03/Oct/2013:13:55:36.111 +0200] "GET /harjoittelu.html HTTP/1.0" 200 2326
```

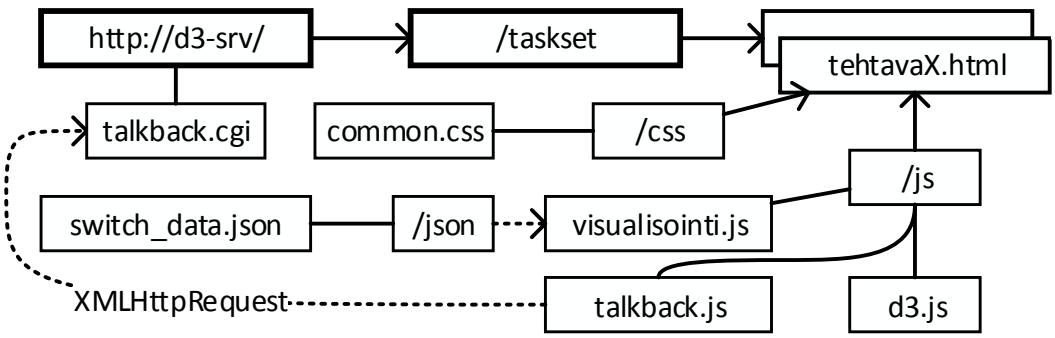
Esimerkissä 192.168.53.131 kertoo kyselyn tehneen asiakaslaitteen IP-osoitteen ja aikaleima kyselyn (3. lokakuuta 2013 kello 13:55:36.111) tarkan ajan millisekunnin tarkkuudella. Lokirivi muodostetaan siinä vaiheessa, kun palvelinprosessi on toteuttanut palvelupyynnön ja vastaus on lähetetty asiakkaalle, eli www-selaimelle.

Visualisoinnin toteuttava JavaScript-koodi jaettiin tiedostoihin, jotka selain lataa harjoittelu- tai tehtävätiedoston määrittelemänä. Selain osaa tarkistaa palvelimelta, onko selaimen pyytämä tiedosto muuttunut. Jos vastauksena on tilakoodi ”304 Not modified”, käytetään välimuistissa olevaa versiota (ks. RFC 2616). Harjoittelutehtävän sisältämä visualisointi oli sama varsinaisten tehtävien kanssa, joten JavaScript-koodi ladattiin muistiin jo harjoittelun yhteydessä. Tällä varmistettiin se, että ensimmäisessä varsinaisessa tehtävässä ei ole muita tehtäviä suurempaa tiedostojen latauksesta aiheutunutta viivettä. On myös oletettavaa, että selain prosessoi samaa JavaScript-koodia yhtä kauan, joten selaimen aiheuttama viive on sama jokaisessa tehtävässä.

Eri tehtävien välisen ajan lisäksi haluttiin tietää tehtävän suorittamiseen tarvittavien osoitusten määrä. Lisäksi käyttäjän toimista haluttiin hienojakoisempaa tietoa jatkotutkimusaiheita ajatellen, mutta esimerkiksi Lazarin ja kumppaneiden (2010) esittelemät instrumentoinnit tuntuivat liian raskailta tarkoitukseen, joten instrumentointia varten päätettiin kehittää yksinkertainen lisäosa. Lisäosa koostui selaimeen tehtävätiedoston yhteydessä ladattavasta JavaScript-moduulista, josta käytettiin nimitystä Talkback (talkback.js), sekä www-palvelimella http-palvelinprosessin kautta toimivasta sovelluksesta (talkback.cgi), joka vastaanotti Talkback:n lähettämiä lokiviestejä.

Talkback hyödyntää W3C-konsortion standardoimaa XMLHttpRequest-objektia (http://www.w3.org/TR/XMLHttpRequest/). XMLHttpRequest-objektin avulla selain voi kommunikoida asynkronisesti esimerkiksi www-palvelimen kanssa. Tällöin esimerkiksi lisätiedon lataus tai jonkin tiedon lähetys ei edellytä sivun uudelleenlatausta, vaan vastaanotto tai lähetys tapahtuu ikään kuin selainprosessin tausta-ajona. Niinpä esimerkiksi lähetyksen kuittauksen odottelu ei vaikuta selaimen käyttöliittymän toimintaan mitenkään ja Talkback:n tapauksessa useita lokiviestejä voi olla jonossa lähetystä odottamassa. Tällä järjestelyllä pyrittiin saavuttamaan se etu, että esimerkiksi verkon viiveet tai mahdolliset ongelmat informaatiota vastaanottavassa palvelinprosessissa eivät aiheuta tiedon katoamista eivätkä muutoksia mittauksessa.

Talkback lähettää tilaviestit samalle palvelimelle, jolta varsinaiset tehtävät ladataan. Vastaanottoa varten kehitettiin yksinkertainen sovellus (talkback.cgi), joka ottaa vastaan JSON-muotoisen objektin ja tallentaa aikaleiman sekä objektin erilliseen tiedostoon. Tällä menettelyllä pyrittiin mahdollisimman yksikäsitteiseen ja tarkkaan ajan mittaamiseen. Yksittäiseen tehtävätiedostoon liittyvät tiedostot, tiedonhaku ja tilaviestin lähetys on esitelty kuvassa 16.



Kuva 16. Tiedostojen keskinäinen sijainti.

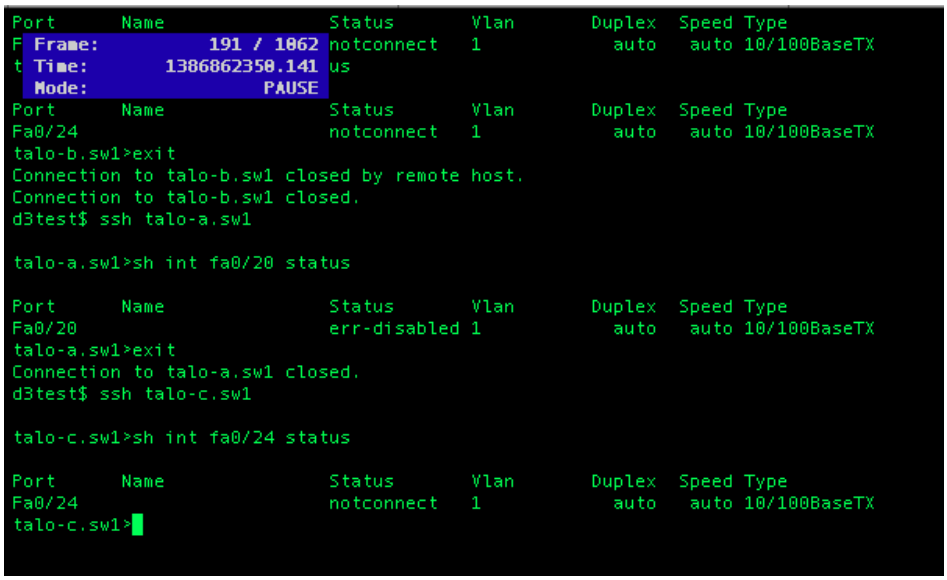
Talkback määriteltiin käynnistymään, kun tehtäväsivu on ladattu (window.onload). Käynnistyessään Talkback lähettää mittauspalvelimelle tilaviestin, jossa se raportoi selainikkunan koon, ladatun sivun osoitteen ja selaimen kellonajan millisekunnin tarkkuudella. D3 mahdollistaa objekteihin liitetyt funktiokutsut, joten solmujen osoituksiin liitettiin Talkback-funktiokutsu. Tällöin Talkback lähettää lokiviestin jokaisesta hiiren osoituksesta. Esimerkki lokiviesteistä (sivun lataus ja hiiren osoitus) on alla:

```
2013/12/12 13:09:16 INFO> {"status":"dokumentti_ladattu","window_innerwidth":1916,"window_innerheight":1091,
"document_url":"http://d3-srv/harjoittelu/", "time":1386846556738}
2013/12/12 13:11:29 INFO> {"status":"click","title":"testi-c.aggr-sw","time":1386846689934}
```

Koko mittauksen sisältävästä tiedosta on mahdollista selvittää yksittäiset osoitukset, jolloin voidaan selvittää kuhunkin tehtävään liittyvien osoitusten kokonaismäärä. Virheellisten osoitusten määrä saadaan vähentämällä tehtäväkohtaisesta osoitusten kokonaismäärästä tehtäväkohtainen osoitusten optimaalinen lukumäärä, jolloin voidaan vertailla tekstipohjaisen käyttöliittymän virhemääriä graafisen käyttöliittymän virhemääriin.

4.5.2. Tekstipohjainen käyttöliittymä

Tekstipohjaisen käyttöliittymän mittauksissa nauhoitettiin osallistujan koko komentorivi-istunto. Istunnon nauhoitus käynnistyi, kun osallistuja valitsi työpöydältä tekstipohjaisen käyttöliittymän. Nauhoitukseen käytettiin avoimen lähdekoodin sovellusta nimeltä *ttyrec* (http://0xcc.net/ttyrec/). Ttyrec toimii kuten unix-työkalu *script*, eli se tallentaa käyttäjän näppäinpainallukset sekä ruudulla näkyvän informaation. Ttyrec valittiin siitä syystä, että se oletuksena tallentaa nauhoitustiedostoon myös aikaleimat[1]. Nauhoitusten katselemiseen on kehitetty avoimen lähdekoodin sovellus nimeltään IPBT (http://www.chiark.greenend.org.uk/~sgtatham/ipbt/), jota hyödynnettiin istuntonauhoitusten katselussa (ks. kuva 17). Kuvassa näkyvien aikaleimojen avulla tehtävien tarkka suoritusaika saatiin kirjattua taulukkoon.



Kuva 17. Esimerkki tekstipohjaisen käyttöliittymän nauhoitteen läpikäymisestä IPBT-sovelluksen avulla.

Tehtäväkohtaisten suoritusaikojen lisäksi nauhoitustiedostoista haettiin kirjoitusvirheiden lukumäärät. Tätä varten toteutettiin yksinkertainen sovellus käyttämällä perl-ohjelmointikieltä (http://www.perl.org) ja Term::TtyRec-moduulia (http://search.cpan.org/~miyagawa/Term-TtyRec-0.03/lib/Term/TtyRec.pm). Sovelluksen toimintaperiaate on yksinkertainen: nauhoitustiedosto käydään läpi silmukassa ja TtyRec-moduulin avulla luetaan käyttäjän näppäinpainallukset. Kun käyttäjän varsinaisen tehtäväosuuden aloitus- ja lopetusajankohdat olivat tiedossa, silmukka saatiin kohdistettua oikein. Silmukan sisäosa etsi askelpalautin -merkkiä. Lopullinen askelpalautinten määrä on siis selvillä nauhoituksen lopussa. Tehtäväkohtaisia kirjoitusvirhemääriä ei mitattu.

Tekstipohjaisessa käyttöliittymässä kytkimet oli nimetty pelkillä laitteen nimillä nimen ja aluenimen yhdistelmän eli absoluuttisen osoitteen (fully qualified domain name; ks. RFC1034)

[1] Optio on olemassa myös *script*-komennossa, mutta aikaleimat sijaitsevat eri tiedostossa.

sijaan. Toisin sanoen kirjautumisessa käytettävä osoite on esimerkiksi muotoa *talo-a.sw1* muodon *talo-a.sw1.organisaatio.fi* sijaan. Myös kirjautumiseen käytettävä käyttäjätunnus ja salasana on tallennettu asetuksiin. Tällöin käyttäjän ei tarvitse erikseen antaa kirjautumisessa käytettävää käyttäjätunnusta, vaan se haetaan asetustiedostosta. Mainitut asetukset ovat luonteeltaan sellaisia perusasetuksia, joita päivittäistä ylläpitotyötä tekevä on todennäköisesti tallentanut itselleen.

Mittaukseen saatavilla olevien verkkolaitteiden lukumäärä oli rajoitettu. Toisaalta tekstipohjaiseen käyttöliittymään liittyvissä tehtävissä käytettiin vain yhtä laitetta kerrallaan, joten tehtävän vaikeus ei muutu laitteiden lukumäärän mukaan. Tällöin samoja laitenimiä voitiin hyödyntää useammassa tehtävässä. Graafisen käyttöliittymän tehtävät kuitenkin alkoivat perustilanteesta (ks. kuva 10), joten tehtävän alkutilanne täytyi saada aikaan myös tekstipohjaiseen käyttöliittymään. Tällöin luotiin tilanne, jossa tarve kirjautua tietylle kytkimelle on joka tehtävässä uusi, joten komentorivihistoria ja rivipuskuri (eli ruudun vieritys taaksepäin) poistettiin käytöstä. Näin tehtävän suoritus alkaa joka tehtävän jälkeen alusta, eikä aiemmin haettu tieto, kuten kytkinporttilistaukset, ole käytettävissä.

4.6. Tehtävät ja mittauksen kulku

Koshmanin (2004) tutkimus otettiin pohjaksi mittauksen toteutukselle. Mittauksen tarkoitus oli selvittää, miten nopeasti käyttäjä löytää tietyn kytkinportin ja miten hyvin käyttöliittymä soveltuu kytkinporttitason visualisointiin. Navigoinnin apukeinona osallistujalla oli mahdollisuus hyödyntää selaimen etsi-toimintoa ja tästä myös päätettiin ohjeistuksessa kertoa. Tutkielman mittausaineistoon valittiin ylätasolle kuusi tietoverkon runkolaitetta kuvaavaa solmua (ks. kuva 10). Tarkoitus oli simuloida oikeaa runkoverkkoa, joten runkolaitteet nimettiin talojen mukaan (ks. kuva 8). Runkolaitteilla oli puolestaan yhdestä viiteen kappaletta jälkeläissolmuja, joilla simuloitiin talojen kerrosjakamoja. Kussakin kerrosjakamossa oli yhdestä viiteen kappaletta kytkimiä. Tehtäviin valituissa kytkimissä oli noin 50 kappaletta jälkeläissolmuja, jotka esittivät kytkinportteja.

Mittauksen osallistujat suorittavat 21 tehtävää per käyttöliittymä, eli yhteensä 42 tehtävää. Tehtävät ovat Koshmanin (2004) tutkimuksen tavoin laadittu siten, että sama tieto oli saatavissa molemmilla käyttöliittymätyypeillä eli tehtävät olivat lähes samat molemmissa käyttöliittymämittauksissa. Järjestelyllä pyrittiin siihen, että tehtäviin kulunut aika olisi mahdollisimman pitkälle vertailukelpoista käyttöliittymien välillä. Käyttöliittymien järjestys arvottiin siten, että puolet osallistujista suoritti ensin tekstipohjaisen käyttöliittymän tehtävät ja puolet graafisen käyttöliittymän tehtävät.

Ennen mittauksen aloittamista osallistujaa pyydettiin täyttämään taustatietolomake (ks. liite 1). Lomakkeen täytön jälkeen osallistujalle esiteltiin molemmat käyttöliittymät (ks. liite 2). Esittely tapahtui niin, että osallistujan käyttöön annettiin koko mittauksen ajaksi tekstipohjaisen käyttöliittymän komentoesimerkkejä esittelevä ohje, joka käytiin läpi ennen mittauksen aloittamista. Ohjeessa oli lyhyt kuvaus myös graafisen käyttöliittymän visualisoimasta tietorakenteesta sekä ohjeistus käyttää selaimen omaa etsi-toimintoa.

Osallistujalle annettiin mahdollisuus harjoitella tekstipohjaisen ja graafisen käyttöliittymän käyttöä ennen mittauksen aloittamista. Kaikki osallistujat harjoittelivat graafisen käyttöliittymän käyttöä. Kaikki noviisiryhmän osallistujat harjoittelivat tekstipohjaista käyttöliittymää. Eksperttiryhmän osallistujista 60 prosenttia tunsi tekstipohjaisen käyttöliittymän entuudestaan niin hyvin, että he eivät pitäneet harjoittelua tarpeellisena.

Tehtävien suoritus alkoi siten, että harjoitteluvaiheen lopuksi osallistujalle annettiin tehtäväpaperi ja lupa alkaa suorittaa tehtäviä tehtäväpaperin mukaisessa järjestyksessä. Tehtävissä osallistujaa pyydettiin merkitsemään tehtäväpaperiin annetun kytkimen ja kytkinportin tila. Kytkinportin tila saattoi olla joko ”connected”, ”not connected” tai ”err-disabled” (ks. liite 3). Ero tehtäväpapereiden välillä oli se, että kun tekstipohjaisessa käyttöliittymässä käytettiin laitteistosta johtuvista syistä kolmea eri verkkolaitetta (ks. alakohta 4.5.2), graafisessa käyttöliittymässä tehtävät kohdistuivat yhteensä seitsemään verkkolaitteeseen.

Kunkin käyttöliittymäosion jälkeen osallistujaa pyydettiin täyttämään käyttäjätyytyväisyyslomake. Mittauksen lopuksi osallistujaa haastateltiin lyhyesti puoliavoimella haastattelulla. Haastattelun kaksi teemaa olivat ”Mitä mieltä olet siitä, kumpi käyttöliittymä sopi paremmin käsillä olleeseen tehtävään?” ja ”Voiko mielestäsi graafisella käyttöliittymällä suorittaa tämänkaltaisia tehtäviä yhtä tehokkaasti, kuin tekstipohjaisella käyttöliittymällä?” Ensimmäisen teeman tarkoituksena oli kartoittaa sitä, mitkä tekijät vaikuttivat osallistujan käyttöliittymävalintaan. Toisen teeman tarkoituksena oli kartoittaa yleisellä tasolla sitä, millainen olisi osallistujan mielestä ihanteellinen graafinen käyttöliittymä ja toisaalta sitä, millaisia edellytyksiä graafisella käyttöliittymällä ylipäätään on korvata olemassa oleva tekstipohjainen käyttöliittymä.

Mittausprosessi esitestattiin ennen varsinaista käyttöä. Esitestaus toteutettiin samalla tavoin, kuten varsinainen mittaus oli tarkoitus toteuttaa, jolloin testauksessa ilmenneet ongelmat tai puutteet saataisiin korjatuksi ennen varsinaista mittausta. Esitestauksessa ei kuitenkaan havaittu ongelmia, joten mittausproseduuri otettiin sellaisenaan käyttöön.

4.7. Aineiston analyysit

Suoritusaikojen analyysi aloitettiin tarkistamalla tehtävälomakkeet ja oikeat vastaukset. Mittauksen raakamuotoista materiaalia alettiin työstää osallistujittain kirjaamalla kunkin tehtävän aloitusaika taulukkoon. Aloitusaika saatiin käyttöliittymäkohtaisesti joko lokitiedostosta tai komentotallenteesta. Kun aloitusajat kaikista tehtävistä oli molempien käyttöliittymien osalta kirjattu, laskettiin aloitusaikojen erotuksena tehtäväkohtainen suoritusaika. Tällä tavoin tehtävän suoritusaika on riippumaton siitä, täyttikö osallistuja tehtäväpaperia tehtävän aikana, eli heti oikean vastauksen löydyttyä, vai vasta tehtävän jälkeen.

Suoritusajoista laskettiin osallistujakohtaiset keskiarvot kummallekin käyttöliittymälle. Aineistolle laskettiin *keskiarvon keskivirhe* (Standard error of the mean, S.E.M.), jotta voidaan

arvioida sitä, kuinka paljon otoksen keskiarvo poikkeaa koko populaation keskiarvosta. S.E.M. on raportoitui kaikkien suoritusaikojen yhteydessä.

Suoritusaikojen tilastollinen analyysi aloitettiin käyttäen 2 x 2 (käyttöliittymä x osallistujaryhmä) *toistettujen mittausten varianssianalyysiä* (repeated measures analysis of variance, ANOVA). Varianssianalyysin jälkeen vertailtiin käyttöliittymiin liittyviä suoritusaikoja *toistettujen mittausten t-testillä* (repeated measures t-test): ensin yleinen käyttöliittymäkohtainen vertailu (tekstipohjainen käyttöliittymä vs. graafinen käyttöliittymä) ja tämän jälkeen osallistujaryhmäkohtainen vertailu (ekspertit: graafinen käyttöliittymä vs. tekstipohjainen käyttöliittymä, noviisit: graafinen käyttöliittymä vs. tekstipohjainen käyttöliittymä). Lisäksi vertailtiin osallistujaryhmiä (noviisit vs. ekspertit) keskenään käyttöliittymäkohtaisesti *riippumattomien otosten t-testillä* (independent measures t-test).

Tekstipohjaisen käyttöliittymän kohdalla askelpalauttimen (backspace) painallusmäärästä laskettiin osallistujakohtainen keskiarvo. Graafisen käyttöliittymän osalta kerättiin osoitusten lukumäärä tehtävittäin. Osoitusten lukumäärästä vähennettiin kunkin tehtävän osalta optimaalinen osoitusmäärä, eli pienin mahdollinen osoitusmäärä, jolla tehtävä oli suoritettavissa. Näin saadut osoitusten lukumäärät kuvaavat virheosoitusten määrää. Osoitusten lukumääristä laskettiin osallistujakohtaiset keskiarvot.

Virheiden lukumääriä käyttöliittymien välillä vertailtiin ensin *toistettujen mittausten t-testillä*. Tämän jälkeen osallistujaryhmäkohtaisia määriä vertailtiin käyttöliittymittäin (graafinen käyttöliittymä: noviisit vs. ekspertit, tekstipohjainen käyttöliittymä: noviisit vs. ekspertit) *riippumattomien otosten t-testillä*.

Käyttäjätyytyväisyyden arvioinnit kirjattiin taulukkoon asteikko- ja käyttöliittymäkohtaisesti. Analysoinnin ensimmäisessä vaiheessa oli tarkoitus löytää ne mahdolliset asteikot, joissa käyttäjätyytyväisyys eroaa käyttöliittymien välillä. Käyttäjätyytyväisyyden tilastolliset analyysit toteutettiin ei-parametrisilla analyysimenetelmillä. Vertailu toteutettiin siten, että asteikko kerrallaan vertailtiin käyttäjätyytyväisyyttä käyttöliittymien välillä *Wilcoxonin etumerkillisten sijalukujen testi* (Wilcoxon matched-pairs signed-ranks test). Tämän jälkeen pyrittiin selvittämään käyttöliittymittäin, poikkeavatko käyttäjäryhmien arvot toisistaan. Vertailu toteutettiin *Mann-Whitneyn U -testillä*.

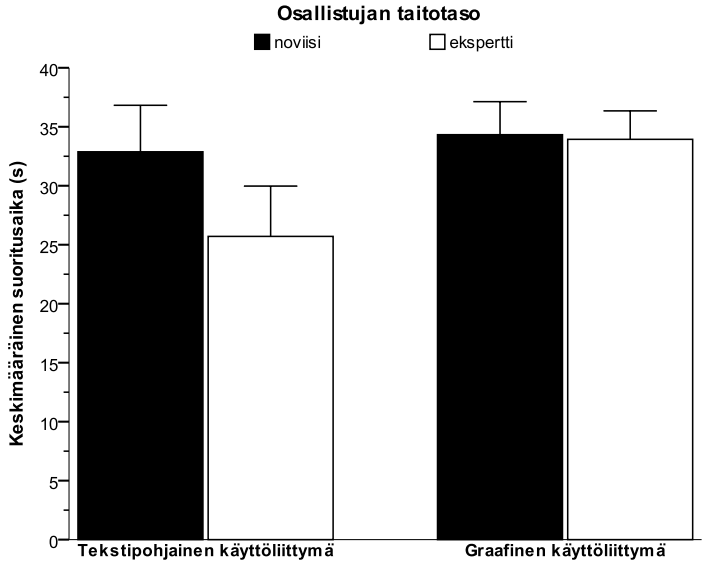
5. Tulokset

Tässä luvussa raportoidaan käytettävyysmittauksen tulokset. Raportointi jakautuu kolmeen kohtaan mittauksen osa-alueiden mukaisesti. Kohdassa 5.1 esitellään ensin navigointitehtävien suoritusaikojen vertailut. Kohdassa 5.2 esitellään käyttäjätyytyväisyyskyselyn tulokset ja niiden analyysi. Kohdassa 5.3 esitellään lyhyesti haastattelujen tuloksia.

Yleisesti ottaen mittaukseen kokonaiskesto oli 45–60 minuuttia per osallistuja. Vaihtelu syntyi lähinnä haastattelujen keston vaihtelusta. Vastausten osalta yhtäkään tehtävää ei jätetty kesken. Tehtävien vastauksissa ei myöskään havaittu virheitä.

5.1. Ajankäyttö navigointitehtävissä

Suoritusaikojen analysoinnin yhteydessä paljastui, ettei kummankaan käyttöliittymän osalta viimeisen tehtävän suoritusaikaa voitu mitata luotettavasti. Tämä mahdollisesti johtui siitä, että tehtäväpaperi annettiin osallistujalle kerralla. Ongelma ilmeni siten, että tekstipohjaisen käyttöliittymän tapauksessa harva osallistuja sulki komentorivi-ikkunan ja graafisen käyttöliittymän tapauksessa siirtyi mittauksen päättävään tehtävään. Niinpä viimeisen tehtävän suoritusajat jätettiin huomiotta kummankin käyttöliittymän osalta. Osallistuja- ja käyttöliittymäryhmittäin jaetut suoritusajat ja keskiarvon keskivirheet (S.E.M.) on esitelty kuvassa 18. Kuvasta 18 nähdään, että osallistujan taitotasolla voisi olla merkitystä tekstipohjaisen käyttöliittymän osalta siten, että ekspertit olivat nopeampia tekstipohjaisen käyttöliittymän kanssa. Sen sijaan graafisen käyttöliittymän osalta taitotasolla ei näyttänyt olevan vaikutusta.



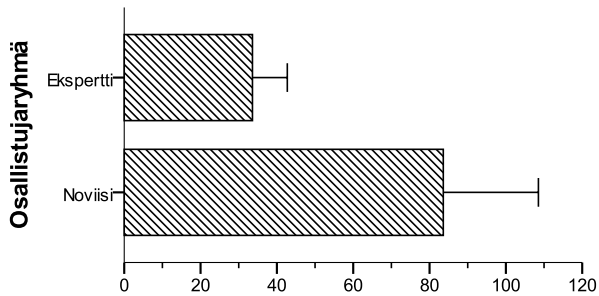
Kuva 18. Suoritusajat käyttöliittymittäin (± 1 S.E.M.).

Tilastollisten analyysien riskitasoksi valittiin 0,1. Aineisto analysoitiin ensin 2 x 2 (käyttöliittymä x osallistujaryhmä) *toistettujen mittausten varianssianalyysillä* (ANOVA). Käyttöliittymäkohtaisten suoritusaikojen (graafinen, tekstipohjainen) välillä havaittiin päävaikutus

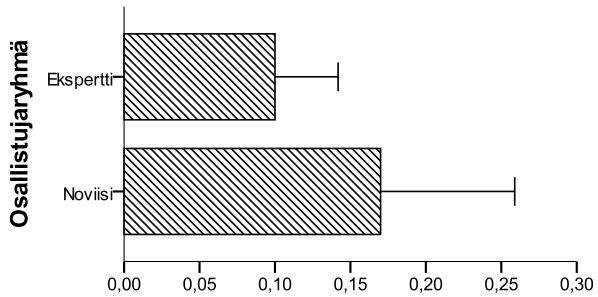
$F(1, 8) = 3{,}6$, $p = 0{,}096$. Osallistujaryhmän ja käyttöliittymän välillä ei havaittu tilastollisesti merkitsevää yhteisvaikutusta ($F(1, 8) = 1{,}8$, $p = 0{,}223$). *Toistettujen mittausten t-testin* tuloksena havaittiin käyttöliittymien välinen ero tilastollisesti melkein merkitseväksi ($t(9) = 1{,}8$), $p = 0{,}103$.

Kokonaisuuden analysoinnin jälkeen siirryttiin vertailemaan ryhmäkohtaisia suoritusaikoja käyttöliittymien välillä. Eksperttiryhmällä *toistettujen mittausten t-testin* tulos oli tilastollisesti merkitsevä ($t(4) = 4{,}2$, $p = 0{,}014$). Noviisiryhmän suoritusajoissa ei havaittu tilastollisesti merkitsevää eroa ($t(4) = 0{,}3$, $p = 0{,}775$).

Suoritusaikojen vertailun jälkeen siirryttiin vertailemaan virheiden lukumääriä käyttöliittymittäin. Tekstipohjaisessa käyttöliittymässä näppäilyvirheiden lukumäärän keskiarvo ± S.E.M. on esitetty kuvassa 19. Graafisessa käyttöliittymässä optimaalisesta osoitusmäärästä poikkeavien osoitusten lukumäärän keskiarvo ± S.E.M. on esitetty kuvassa 20. Kuvasta 19 voidaan nähdä, että osallistujan taitotasolla voisi olla vaikutusta näppäilyvirheiden lukumäärään siten, että ekspertit tekivät noviiseja vähemmän näppäilyvirheitä. Myös kuvasta 20 voidaan havaita, että osallistujan taitotaso saattaa vaikuttaa optimaalisesta osoitusmäärästä poikkeavien osoitusten lukumäärään. Tämä näkyy siten, että eksperttien optimaalisesta osoitusmäärästä poikkeavien osoitusten lukumäärä on noviiseihin verrattuna lähempänä optimaalista osoitusmäärää. Osallistujaryhmällä siis saattaisi olla vaikutusta virheiden lukumäärään käyttöliittymästä riippumatta siten, että ekspertit tekevät noviiseja vähemmän osoitus- tai näppäilyvirheitä.



Kuva 19. Näppäilyvirheiden lukumäärän keskiarvo (± 1 S.E.M.) tekstipohjaisessa käyttöliittymässä osallistujaryhmittäin.

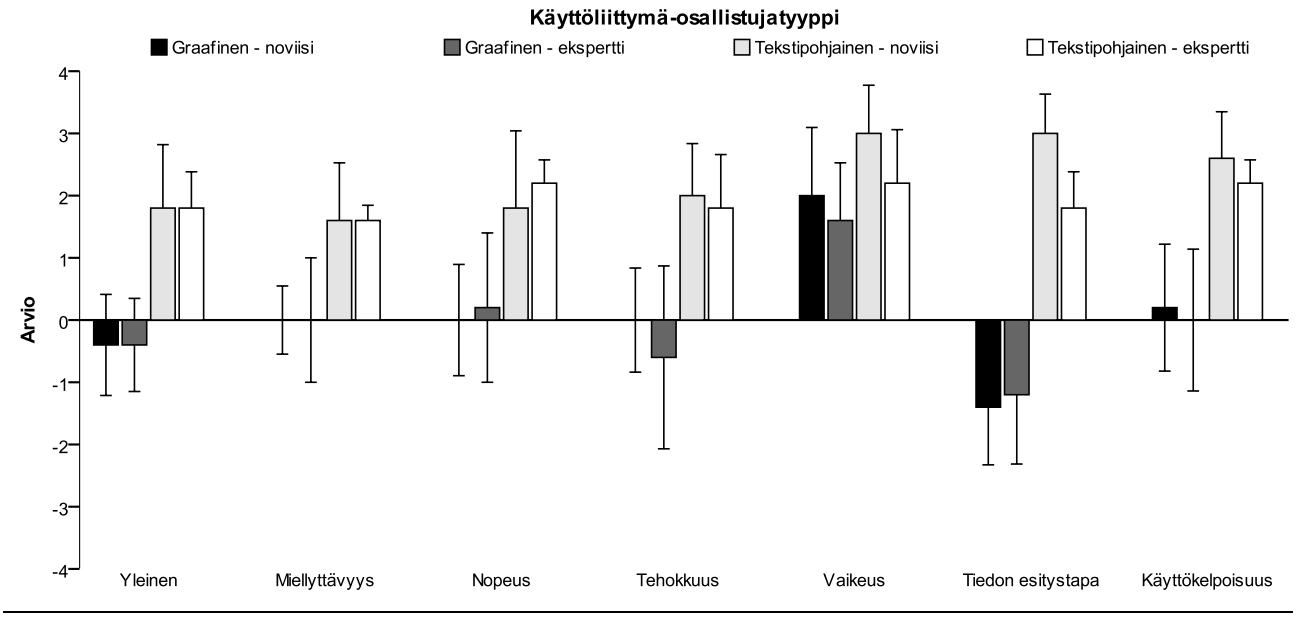


Kuva 20. Optimaalisesta poikkeavien osoitusten lukumäärän keskiarvo (± 1 S.E.M.) graafisessa käyttöliittymässä osallistujaryhmittäin.

Virheiden lukumääriä vertailtiin ensin käyttöliittymäkohtaisesti (graafinen vs. tekstipohjainen) *toistettujen mittausten t-testillä*, jonka tulos oli tilastollisesti merkitsevä ($t(9) = 3{,}9$, $p = 0{,}004$). Osallistujaryhmiä vertailtiin vielä käyttöliittymäkohtaisesti *riippumattomien otosten t-testillä*. Tekstipohjaisessa käyttöliittymässä ero osallistujaryhmien välillä oli tilastollisesti merkitsevä ($t(8) = 1{,}9$, $p = 0{,}096$). Graafisessa käyttöliittymässä ero ei ollut tilastollisesti merkitsevä ($t(8) = 1{,}4$, $p = 0{,}496$).

5.2. Käyttäjätyytyväisyys

Käyttäjätyytyväisyysmittauksen ($n = 10$) tulokset on esitetty kuvassa 21. Kuvasta 21 nähdään, että käyttöliittymä saattaisi vaikuttaa käyttäjätyytyväisyyteen siten, että osallistujat ovat tyytyväisempiä tekstipohjaiseen käyttöliittymään.



Kuva 21. Käyttäjätyytyväisyysarviointi käyttöliittymä- ja osallistujatyypeittäin (± 1 S.E.M.).

Käyttäjätyytyväisyyttä analysoitiin *Wilcoxonin etumerkillisten sijalukujen testillä* asteikko kerrallaan. Tilastollisen analyysin tulokset on esitetty taulukossa 1. Taulukosta 1 voidaan nähdä, että käyttöliittymällä näyttäisi olevan vaikutusta käyttäjätyytyväisyyteen siten, että asteikoilla yleinen, tiedon esitystapa ja käyttökelpoisuus käyttöliittymien välinen ero oli tilastollisesti merkitsevä.

Käyttäjätyytyväisyyden analysointia jatkettiin asteikko kerrallaan siten, että *Mann-Whitneyn U -testillä* vertailtiin osallistujaryhmien käyttäjätyytyväisyyttä käyttöliittymäkohtaisesti. Osallistujaryhmien käyttäjätyytyväisyydessä ei havaittu tilastollisesti merkitseviä eroja kummankaan käyttöliittymän osalta.

Asteikko	*Z*-testisuureen arvo
Yleinen	2,0, $p = 0,04$
Miellyttävyys	2,0, $p = 0,05$
Nopeus	1,7, $p = 0,09$
Tehokkuus	1,8, $p = 0,07$
Vaikeus	1,0, $p = 0,30$
Tiedon esitystapa	2,5, $p = 0,01$
Käyttökelpoisuus	2,0, $p = 0,04$

Taulukko 1. Käyttäjätyytyväisyysanalyysin tulos käyttöliittymien välillä.

5.3. Haastattelut

Haastatteluissa osallistujalta kysyttiin ensin sitä, kumpi käyttöliittymä sopi paremmin käsillä olleeseen tehtävään. Osallistujista 80 prosenttia koki tekstipohjaisen käyttöliittymän helpommaksi ja nopeammaksi tavaksi suorittaa annetut tehtävät. Pääasiallinen syy oli kokemus siitä, että järjestelmän tila on paremmin selvillä tekstipohjaisessa käyttöliittymässä. Toinen pääsyy liittyi siihen, että osallistujat kokivat tekstipohjaisen käyttöliittymän tutummaksi ja tehokkaammaksi tavaksi suorittaa annetut tehtävät.

Toinen haastattelukysymys koski sitä, voiko osallistujan mielestä graafisella käyttöliittymällä suorittaa tämänkaltaisia tehtäviä yhtä tehokkaasti, kuin tekstipohjaisella käyttöliittymällä. Osallistujista 30 prosenttia oli sitä mieltä, että graafinen käyttöliittymä voi olla vähintään yhtä tehokas. Perustelut liittyivät siihen, että osallistujat ajattelivat suoritusta osana ylläpitotyötä: vaikka graafinen käyttöliittymä on yksittäisessä mekaanisessa toimenpiteessä hitaampi, nopeus tulee esille silloin, kun tehtävät liittyvät toisiinsa. Osallistujat antoivat esimerkkejä tehtäväkokonaisuuksista, joihin liittyi verkkolaitteen nimen selvittäminen sijainnin perusteella, kirjautuminen ja erilaiset yhdistetyt tiedonhakutoiminnot, joita graafisella käyttöliittymällä voi toteuttaa samalla kertaa.

Yleisesti ottaen 80 prosenttia mittaukseen osallistuneista mainitsi graafisen käyttöliittymän suurimmaksi ongelmaksi sen, että pilvimuodostelmasta etsiminen oli epäintuitiivista. Tämä johtui siitä, että solmujen piirtojärjestys oli satunnainen, jolloin solmuilla ei ollut keskinäistä järjestystä. Lisäksi solmun jälkeläissolmut aukesivat ruudulla kohtaan, jossa oli tilaa, eivätkä esimerkiksi suoraan edeltäjänsä viereen. 40 prosenttia koki tämän ominaisuuden ennustettavuuden puuttumisena. Niinikään 40 prosenttia koki, että solmujen järjestäytyminen kesti liian kauan, joten solmujen olisi pitänyt joko vakiinnuttaa sijaintinsa nopeammin tai ensin vakiinnuttaa sijaintinsa ja piirtyä vasta sitten ruudulle.

50 prosenttia osallistujista mainitsi hämääntyneensä siitä, että vain yhden jälkeläissolmun sisältävä solmu aukeaa myös jälkeläissolmun osalta (esimerkki tällaisesta tilanteesta on esitetty kuvassa 11). Ominaisuus koettiin käyttöliittymän epäjatkuvuutena.

6. Tulosten pohdintaa

Tilastollisissa analyyseissä hylkäysvirheestä johtuvan virhepäätelmän (tyypin I virhe) riski oli noin 10 prosenttia. Aineisto oli kuitenkin hyvin pieni, jolloin hylkäysvirheen riskirajana pidetty 5 prosenttia vastaisi puolikasta ja 10 prosenttia yhtä mittaukseen osallistujaa. Tämän perusteella tuloksia lienee joka tapauksessa syytä käsitellä suuntaa antavina.

Tehtävät (ks. liite 3) oli tarkoituksella laadittu yksinkertaisiksi. Tausta-ajatuksena oli se, että alkutilanne, jossa verkko ei toimi, tai lähituki haluaa tarkistaa tietyn kytkinportin tilan, on sama käyttöliittymästä riippumatta. Alkutilanteen selvitystyö on molempien käyttöliittymien osalta sama, jolloin tehtävän alussa kysytty kytkinportti on selvillä. Tavoitteena oli vertailla sitä, miten nopeasti tietty tieto on löydettävissä ja onko tiedonhakunopeudessa eroja käyttöliittymien välillä. Aineistosta olisi voinut analysoida tarkemmin myös sitä, onko esimerkiksi virhetilassa olevan kytkinportin havaitsemisnopeudessa eroja käyttöliittymien välillä, mutta tilastollinen analyysi rajattiin tämän tutkielman ulkopuolelle.

Navigointitehtävien ajankäytön 2 x 2 (käyttöliittymä x osallistujaryhmä) varianssianalyysissä havaittiin käyttöliittymän merkitsevä päävaikutus. Päävaikutus käy ilmi yhdistetyistä suoritusajoista: suoritetuissa mittauksissa tekstipohjainen käyttöliittymä oli 4,8 sekuntia graafista käyttöliittymää nopeampi. Suoritusaikojen eron olisi voinut odottaa olevan päinvastainen (ks. esim. Rauterberg, 1992).

Jatkoanalyysin tuloksena havaittiin tilastollisesti merkitsevä ero eksperttiryhmällä graafisen ja tekstipohjaisen käyttöliittymän välillä. Eksperttiryhmällä keskimääräinen ero tehtäväkohtaisessa suoritusajassa oli 8,2 sekuntia tekstipohjaisen käyttöliittymän eduksi. Noviisiryhmässä tilastollisesti merkitsevää eroa ei havaittu ja keskimääräinen suoritusaikaero käyttöliittymien välillä oli 1,4 sekuntia tekstipohjaisen käyttöliittymän ollessa nopeampi.

Eksperttiryhmä oli molemmissa käyttöliittymissä noviisiryhmää nopeampi. Graafisessa käyttöliittymässä keskimääräinen ero suoritusajoissa oli 0,4 sekuntia ja tekstipohjaisessa käyttöliittymässä 7,2 sekuntia. Sinänsä tällainen tulos oli odotettu, sillä eksperttiryhmän valintakriteeri oli tekstipohjaisen käyttöliittymän päivittäinen tai lähes päivittäinen käyttö. Tulos on myös yhtenevä Rauterbergin (1992) raportoiman tuloksen kanssa.

Käyttöliittymien välisiä suoritusaikoja vertailtaessa tulee huomioida, että graafisessa käyttöliittymässä laitteiden määrän kasvu lisää samalla solmujen kokonaismäärää. Osa solmuista on jatkuvasti näkyvillä, joten graafisen käyttöliittymän tehtävät vaikeutuvat laitteiden lukumäärän lisääntyessä. Tekstipohjaisessa käyttöliittymässä käytetään vain yhtä laitetta kerrallaan, joten tekstipohjaisessa käyttöliittymässä tehtävän vaikeustaso ei muutu laitteiden määrän lisääntyessä. Toisaalta tekstipohjaisessa käyttöliittymässä esimerkiksi verkkolaitteen nimen tai annettujen käskyjen pituus pidentää suoritusaikaa. Tässä tutkielmassa ei mitattu myöskään esimerkiksi kytkinpinossa toisiin kytkimiin siirtymistä, joilla graafinen käyttöliittymä voisi mahdollisesti kuroa umpeen aikahäviötä.

Graafisen ja tekstipohjaisen käyttöliittymän suoritusaikavertailuissa ei ole todettu yksiselitteistä paremmuutta kummankaan käyttöliittymän osalta. Nyt toteutetun mittauksen suoritusajoista voitaneen todeta se, että tulos on linjassa niiden kirjallisuuden tulosten kanssa, joissa tekstipohjainen käyttöliittymä osoittautui nopeammaksi ja toisaalta ristiriidassa niiden kanssa, joissa graafinen käyttöliittymä oli nopeampi. (ks. esim. Staggers & Kobus, 2000, 174; Chen & Zhang, 2007, 125–126.) Prototyyppikäyttöliittymässä hyödynnettiin selaimen tekstikenttähakua, jolloin graafisella käyttöliittymällä oli myös tekstipohjaisen käyttöliittymän ominaisuuksia. Tekstihakua ei erikseen mitattu, mutta käyttöliittymien osittainen samankaltaisuus voisi selittää sitä, miksei käyttöliittymien välillä havaittu merkitsevää nopeuseroa. Pohdinnan kannalta lienee hedelmällisempää siirtyä tarkastelemaan muita mittauksen tuloksia.

Tekstipohjaisessa käyttöliittymässä noviisit tekivät yleisesti ottaen yli 50 % enemmän näppäilyvirheitä ekspertteihin verrattuna. Ero osallistujaryhmien välillä oli tilastollisesti merkitsevä 6,7 prosentin riskitasolla. Toisaalta molemmilla osallistujaryhmillä oli lähes yhtä pitkä kokemus tekstipohjaisen käyttöliittymän käytöstä (ks. kohta 4.2). Keskiarvon keskivirhe (S.E.M.) oli noviisiryhmällä yli 2,5-kertainen eksperttiryhmään nähden, joten pienen otoskoon vuoksi noviisiryhmän tulos saattoi johtua myös yksittäisten osallistujien näppäilyvirhemääristä. Luotettavampi analyysi edellyttäisi isompaa otoskokoa.

Virheiden osalta on huomioitavaa, että asetusmuutoksia tehtäessä verkkolaitteessa tulee useimmiten siirtyä niin sanottuun ylläpitotilaan. Siirtyminen ylläpitotilaan edellyttää ylläpitosalasanan (ns. enable-salasana Cisco IOS:ssa) syöttämistä. Tämä lisää kirjoitettavan tekstin määrää ja voisi vaikuttaa tekstipohjaisen käyttöliittymän suoritusaikoihin. Tarkemmat päätelmät edellyttäisivät osallistujakohtaisten virhemäärien analyysiä. Lopulta aineiston pienuuden vuoksi virhemäärien osallistujaryhmäkohtaisista eroista ei liene järkevää päätellä tämän enempää.

Virheiden osalta tulee huomioida myös se, että kyse oli mittaustilanteesta. Huomioitavaa on, että mittaustilanteen paine ei tuntunut vaikuttavan samankaltaisesti graafisessa käyttöliittymässä. Ensinnäkin käyttöliittymien välisten virhemäärien analyysissä havaittiin tilastollisesti merkitsevä ero käyttöliittymien välillä. Keskimääräinen virheosoitusten lukumäärä graafisessa käyttöliittymässä oli vähemmän kuin yksi, joten voitaneen todeta, että tekstipohjainen käyttöliittymä on graafista käyttöliittymää virhealttiimpi. Toisekseen graafisessa käyttöliittymässä virheiden lukumäärien ero käyttäjäryhmien välillä ei ollut tilastollisesti merkitsevä (joskaan tilastollinen merkitsevyys tekstipohjaisen käyttöliittymän osalta ei ollut itsestään selvä). Osallistujaryhmät siis suoriutuivat tehtävistä kutakuinkin samankaltaisilla virhemäärillä.

Graafinen käyttöliittymä osoittautui vähemmän virhealttiiksi käyttöliittymäksi. Tilastollisesti merkitsevä ero virheiden määrässä käyttöliittymien välillä tuli esille myös Staggersin ja Kobusin (2000) tutkimuksessa. Käyttöliittymien välinen keskimääräinen suoritusaikaero oli kuitenkin päinvastainen virheiden lukumäärään nähden, joten käyttöliittymän virhealttius ei ilmeisesti ainakaan suoraan vaikuttanut sen nopeuteen. Myös Rauterberg (1992) toteaa, ettei käyttöliittymien välinen suoritusaikaero selity näppäinpainallusten lukumäärällä. Virheiden määrä graafisessa

käyttöliittymässä oli jo hyvin lähellä nollaa, joten voitaneen sanoa, ettei suoritusaikaeroa saada kavennettua vähentämällä osoitusvirheiden määrää. Toisaalta osoitusvirheiden vähäisyyden perusteella voisi ajatella, että ainakin osa graafisen käyttöliittymän toiminnallisuudesta oli onnistunutta. Tarkemmin graafisen käyttöliittymän onnistumista käsitellään käyttäjätyytyväisyyden analyysin sekä haastattelujen yhteydessä.

Käyttäjätyytyväisyyden osalta tilastollisesti merkitsevä ero havaittiin asteikoilla yleinen, tiedon esitystapa ja käyttökelpoisuus. Kun aiempien analyysien tapaan hyväksytään 10 prosentin riski hylkäysvirheestä johtuvaan virhepäätelmään, merkitsevä ero havaitaan myös asteikoilla miellyttävyys, nopeus ja tehokkuus. Tällöin ainoastaan vaikeusaste arvioitiin samankaltaiseksi molemmissa käyttöliittymissä. Osallistujat eivät tehneet virheitä tehtävien suorituksissa, joten muita mitattavia eroja käyttöliittymien ilmaisuvoimassa ei havaittu.

Graafinen käyttöliittymä sai tässä mittauksessa kautta linjan negatiivisemmat arviot. Haastattelujen perusteella negatiivisen arvion voinee selittää osin osallistujien pettymyksellä käyttöliittymän toimintaan. Osallistujat kokivat käyttöliittymän tekstipohjaista käyttöliittymää kankeammaksi ja hitaammaksi, eli kaiken kaikkiaan graafinen käyttöliittymä ei osallistujien mielestä soveltunut kovinkaan hyvin tehtäväänsä. Haastatteluissa osallistujat mainitsivat solmujen värikoodauksen. Yleisesti ottaen punainen väri koettiin huomiovärinä nousevan muita värejä paremmin esille. Siinä mielessä graafinen käyttöliittymä voisikin soveltua kiinnittämään käyttäjänsä huomio erityisesti virhetilanteissa.

Suurin ero tekstipohjaisen käyttöliittymän hyväksi oli 3,7 yksikköä asteikolla, jossa arvioitiin tiedon esitystapaa. Muista asteikoista käyttökelpoisuus (ero 2,3 yksikköä) ja yleinen (ero 2,2 yksikköä) edustivat suurimpia eroja käyttöliittymien välillä. Haastattelujen perusteella suurin ongelma tiedon esitystapaan liittyen oli visualisointiin liittyvä järjestäytymättömyys. Tekstipohjaista käyttöliittymää pidettiin myös ilmaisuvoimaltaan selkeämpänä ja sen vaste annettuun komentoon oli selkeämmin havaittavissa.

Käyttäjätyytyväisyydessä ei havaittu tilastollisesti merkitseviä eroja osallistujaryhmien välillä, eli käyttäjätyytyväisyys oli hyvin samankaltaista molemmissa osallistujaryhmissä. Tulos, jossa yksi käyttöliittymä sopii molemmille käyttäjäryhmille, saa tukea myös kirjallisuudesta. (Whiteside, Jones, Levy & Wilson, 1985.) Arviointien samankaltaisuus tässä mittauksessa saattaisi osin selittyä sillä, että molemmilla osallistujaryhmillä oli kokemusta tekstipohjaisista käyttöliittymistä keskimäärin yli kymmenen vuoden ajalta.

Graafinen käyttöliittymä sai tässä mittauksessa kautta linjan negatiivisemmat käyttäjätyytyväisyysarviot. Arvioita olisi houkuttelevaa vertailla muihin samankaltaisilla asteikoilla toteutettuihin tutkimuksiin, mutta graafisten käyttöliittymien konteksti- ja käyttäjäsidonnaisuuden vuoksi vertailu ei ole perusteltua. Siten kirjallisuudessa esitellyt, osin ristiriitaisetkin, tulokset eivät ole yleistettävissä nyt suoritetun mittauksen tuloksiin, eikä nyt suoritetun mittauksen tuloksia voi kovin luotettavasti vertailla ilman hyvin samankaltaista käyttäjäryhmää sekä tehtäviä.

Mittauksen mahdollisina virhelähteinä voinee pitää selainversiota, käyttöjärjestelmää sekä syöttölaitteita. Mittauksen aikana sekä haastatteluissa nousi esille se, miten eri selaimissa on sijoiteltu esimerkiksi etsi-toiminto: Mozilla Firefoxissa etsi-toiminto tulee oletuksena alas vasemmalle, Google Chromessa ja Safarissa ylös oikealle. Alun perin tarkoitus oli antaa osallistujan itse valita käytettävä selain. Ennen mittausta kuitenkin havaittiin, että yhdessä D3-grafiikkakirjaston kanssa selaimen etsi-toiminto ei toiminut oikein Safarissa eikä Google Chromessa, jolloin ainoaksi vaihtoehdoksi jäi käyttää Mozilla Firefoxia.

Käyttöjärjestelmän oikotiet selaimessa (esimerkiksi komento-f tuo esille selaimen etsi-toiminnon) tai komentorivityökalun merkit, kuten putki, olivat tutumpia niille osallistujille, jotka olivat tottuneet käyttämään Mac OS X -käyttöjärjestelmää. 40 prosenttia osallistujista piti hiirtä tai näppäimistöä epämukavana käyttää. Epämukavuus liittyi joko hiiren liialliseen herkkyyteen näppäimistön erilaiseen tuntumaan osallistujan normaalisti käyttämiin verrattuna. Pääasiallista käyttöjärjestelmää, suosikkiselainta tai tietoja syöttölaitteista ei kuitenkaan osallistujilta kysytty, joten niiden vaikutusta tulokseen ei tarkemmin voida arvioida. Tätä tutkielmaa varten kehitetyn selaininstrumentoinnin avulla olisi mahdollista toteuttaa laajempi mittaus, jossa osallistujat käyttäisivät henkilökohtaisia työvälineitään graafisen käyttöliittymän osalta. Tekstipohjaisen käyttöliittymän instrumentointiin voisi tällaisessa tapauksessa käyttää esimerkiksi paikallisesti käytettävää komentoemulaattoria.

Graafisen käyttöliittymän etuna on se, että käyttöliittymän käyttöön tarvitaan vain toinen käsi. Tämä helpottaa esimerkiksi tilanteessa, jossa käyttäjä tarvitsee toista kättään esimerkiksi puhelimen kannatteluun. Graafisessa käyttöliittymässä osoitusvirheiden määrä oli merkittävästi pienempi tekstipohjaiseen käyttöliittymään verrattuna, jolloin eräs jatkotutkimusaihe olisi tutkia esimerkiksi puhelinkeskustelun aiheuttaman kognitiivisen kuormituksen mahdollisia vaikutuksia virheisiin ja suoritusaikoihin.

Tutkielman käyttöliittymä kehitettiin työpöytäkäyttöä silmälläpitäen. Yleistyvä mobiilikäyttö antaa aiheen pohtia myös kosketusnäytön avulla toteutettavan ylläpidon erityispiirteitä. Tässä tutkielmassa kehitetty graafinen käyttöliittymä on jatkokehitettävissä siten, että käyttö mobiililaitteilla on mahdollista. Olisikin mielenkiintoista vertailla suoritusaikaeroja esimerkiksi kosketusnäytön näppäimistöllä käytettävän tekstipohjaisen käyttöliittymän ja graafisen käyttöliittymän välillä.

Tuloksia pohdittaessa on otettava huomioon se, että käytetty aineisto oli kooltaan varsin rajoitettu. Molemmille osallistujaryhmille tekstipohjainen käyttöliittymä oli varsin tuttu, joten noviisiryhmän osallistujat eivät välttämättä olleet niin ”noviiseja”, kuin mitä mittauksen suunnitteluvaiheessa tarkoitettiin. Noviisiryhmä olisi ehkä ollut viisaampaa valita suhteessa tekstipohjaisen käyttöliittymän käyttöön. Toisaalta käyttöliittymien välinen suoritusaikaero eksperttiryhmässä havaittiin selvästi, joten lienee selvää, että graafinen käyttöliittymä vaatii vielä jatkokehitystä.

7. Päätelmiä

Tämän tutkielman ensimmäisen osan tavoite oli esitellä erilaisia havaintotiedon lajeja ja tapoja kerätä havaintotietoa. Kerätty havaintotiedon muoto riippuu keruumenetelmästä ja menetelmien erilaisuus aiheuttaa erilaisia haasteita tiedon jalostuksessa. Kenties tästä johtuen tiedon visualisointiin kehitetyt järjestelmät ovat vielä pitkälti painottuneita yksittäisen tietolähteen tuottaman havaintotiedon käsittelyyn.

Erilaisia seurantajärjestelmiä käyttävän henkilön tulisikin olla tietoinen erilaisista havaintotiedon lajeista ja niiden käyttötavoista. Lisäksi ylläpitotyötä tekevän henkilön työmäärää lisää se tosiasia, että ongelmien selvittelyyn tarvitaan mahdollisesti usean eri tavoin toimivan järjestelmän rinnakkainen käyttö. Ensimmäisen osan lopuksi esiteltiin eräs vaihtoehto havaintotiedon visualisointiin. Tutkielmassa haluttiin selvittää, olisiko mahdollista kehittää vastaavalla tavalla toimivaa graafista käyttöliittymää kytkinporttien visualisointiin.

Tutkielman toisessa osassa tarkasteltiin kytkinporttien visualisointia varten kehitettyä graafista käyttöliittymää. Toiminnallisuutta mittaavassa osiossa oli tehty useita oletuksia käyttötilanteesta: esimerkiksi suoritusaikamittauksiin liittyvät kytkimet ja kytkinportit olivat ennakolta tiedossa. Näin ei useinkaan todellisuudessa ole, vaan tieto täytyy ensin etsiä jostain muualta. Tällöin jos tietokannan sisältö olisi visualisoitu graafiseen käyttöliittymään, samaa käyttöliittymää voisi käyttää sekä etsimiseen, että tiettyjen määriteltyjen toimenpiteiden suorittamiseen.

Tässä tutkielmassa graafinen käyttöliittymä osoittautui tekstipohjaista käyttöliittymää hitaammaksi ja osallistujat pitivät sitä epämukavana käyttää. Haastatteluissa osallistujat pitivät tärkeänä ominaisuutena sitä, että käyttöliittymän toiminta on ennustettavaa ja käyttäjällä on ikään kuin varma tieto siitä, miten käyttöliittymä toimii. Voimavektoreihin perustuvassa visualisointimenetelmässä animoitu järjestäytyminen, sattumanvarainen sijainti ja keskinäinen järjestys olivat keskeisiä puutteita. Mittauksen alussa havaittiin myös eroja selainten etsi-toiminnon toteutuksessa. Etsi-toiminto voisi olla järkevää sijoittaa osaksi käyttöliittymää, jolloin toiminnallisuus olisi selainriippumaton ja toiminnallisuuden yhteyteen voisi liittää esimerkiksi tietokantahakuja. Kaiken kaikkiaan nyt kehitettyä visualisointia ei voine pitää sellaisenaan käyttökelpoisena.

Graafisen käyttöliittymän toimintovalikoima on väistämättä rajatumpi tekstipohjaiseen käyttöliittymään verrattuna, mutta ylläpitotyön kannalta keskeiset toiminnot voitaneen helposti sijoittaa myös graafiseen käyttöliittymään. Tällaisen toiminnallisuuden voisi ajatella helpottavan erityisesti niitä, jotka tekevät harvoin ylläpitotyötä. Graafisen käyttöliittymän etuna voinee erityisesti tämän ryhmän osalta pitää sitä, että käyttäjälle ei tarjota käyttötarkoitukseen nähden haitallisia tai tarpeettomia toimintoja – toisin sanoen graafisen käyttöliittymän avulla voisi olla helpompaa estää käyttäjää tekemästä virheitä. Toisaalta oikeat toiminnot vaativat riittävän määrän parametreja, ennen toiminnon toteuttamista. Tällä tavoin käyttäjällä voisi olla käytössään ikään

kuin ylimääräinen virheentarkistuskierros ennen toiminnon suorittamista. Graafisen käyttöliittymän mahdollisuuksia on siis aiheellista tarkastella jatkossakin.

Tässä tutkielmassa toteutetun mittauksen laatuun liittyvä keskeinen heikkous liittyy pienehköön aineistoon, joka altistaa virhepäätelmille. Lisäksi osallistujaryhmänä noviisikäyttäjät eivät tekstipohjaisen käyttöliittymän käyttökokemuksen perusteella eronneet riittävästi eksperttikäyttäjistä. Jatkossa noviisiryhmän valintaperusteisiin tulee kiinnittää tarkempaa huomiota erityisesti tekstipohjaisen käyttöliittymän käyttökokemuksen osalta.

Lienee syytä todeta, että tässä tutkielmassa esiteltyjen havaintotiedon hakumenetelmien avulla ei vielä voida sanoa mitään esimerkiksi suorituskyvystä tai viiveestä, eli siitä, kuinka nopeasti isoja tietoaineistoja voidaan liikutella verkossa tai kauanko paketin matka kestää lähteestä kohteeseen. Toistaiseksi ei osata sanoa myöskään mitään varmaa esimerkiksi joukkolähetyksen (multicast) toimivuudesta tai varsinaisten palvelujen saatavuudesta, eli siitä, kuinka kuormitettuja palvelimet ovat tai miten nopeasti ne vastaavat käyttäjien kyselyihin. Näiden tietojen keruu lisää havaintotiedon lajeja entisestään.

On myös huomattavaa, ettei mikään järjestelmä korvaa kokenutta tulkitsijaa, sillä viime kädessä tulkitsijasta riippuu, miten järjestelmän tuottamaa tietoa käytetään. Tulkitsijan kykyyn tulkita vaikuttaa ainakin toimintaympäristön tuntemus (eli miten valvottava kokonaisuus on suunniteltu, toteutettu ja nimetty), kerätyn havaintotiedon kokonaisvaltaisuus (eli kerätäänkö havaintotietoa yhdenmukaisesti ja riittävän laaja-alaisesti), käsitys hallintajärjestelmän käyttökelpoisuudesta, kokemus ja oppimiskyky.

Osallistujat antoivat haastatteluissa palautetta siitä, miten graafista käyttöliittymää tulisi kehittää palvellakseen paremmin käyttäjien tarpeita. Nyt toteutettua mittausta voisikin pitää käyttöliittymän iteratiivisen kehitysprosessin ensimmäisenä vaiheena. Tutkielman mittausosion avulla saatiin arvokasta käyttökokemusta eri käyttöliittymien instrumentoinnista suoritusajan mittaamiseksi. Tutkielman tarpeisiin kehiteltyjä instrumentointimenetelmiä voidaan jatkossakin hyödyntää tämäntyyppisissä mittauksissa.

Käyttöliittymiä ja työkaluja suunniteltaessa tulisi jatkossakin suunnitella yhdessä järjestelmän tulevien käyttäjien kanssa ja toisaalta antaa käyttäjälle mahdollisuus valita se työväline, jolla kulloinenkin tehtävä on tehokkainta suorittaa. Lopulta ylläpidon tärkein tavoite on taata verkon käyttäjälle sujuva käyttökokemus. Ylläpito on onnistunutta, jos sen läsnäoloa ei huomaa.

Viiteluettelo

Bostock, M., Ogievetsky, V., & Heer, J. (2011). D³ data-driven documents. *IEEE Transactions on Visualization and Computer Graphics, 17*(12), 2301–2309.

Bradley, M. M., & Lang, P. J. (1994). Measuring emotion: The self-assessment manikin and the semantic differential. *Journal of Behavior Therapy and Experimental Psychiatry, 25*(1), 49–59.

CERT-FI. (2010a). *Tietoja varastavat haittaohjelmat.* Saatavilla: http://www.cert.fi/attachments/certtiedostot/5n8rVzH5M/Tietoja_varastavat_haittaohjelmat.pdf (viitattu 6.4.2014).

CERT-FI. (2010b). *Urkittuja sähköpostitunnuksia käytetään roskapostin lähettämiseen.* Saatavilla: http://www.cert.fi/tietoturvanyt/2010/10/ttn201010051541.html (viitattu 6.4.2014).

Chen, J. W., & Zhang, J. (2007). Comparing text-based and graphic user interfaces for novice and expert users. In *AMIA Annual Symposium Proceedings 2007* (pp. 125–129). American Medical Informatics Association.

Chin, J. P., Diehl, V. A., & Norman, K. L. (1988). Development of an instrument measuring user satisfaction of the human-computer interface. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems* (pp. 213–218). Washington, D.C., USA: ACM.

Claise, B., & Wolter, R. (2007). *Network management: Accounting and performance strategies*. Indianapolis, IN: Cisco Press.

CSC – Tieteen tietotekniikan keskus Oy. (2011). *Funet weathermap.* Saatavilla: http://www.csc.fi/funet/status/tools/wm (viitattu 8.12.2011).

Gansner, E. R., Hu, Y., & Kobourov, S. G. (2010). GMap: Drawing graphs as maps. In *Proceedings of the 17th International Conference on Graph Drawing* (pp. 405–407). Chicago, IL, USA: Springer Berlin Heidelberg.

Gansner, E. R., & North, S. C. (2000). An open graph visualization system and its applications to software engineering. *Software: Practice and Experience, 30*(11), 1203–1233.

Liao, Q., Blaich, A., VanBruggen, D., & Striegel, A. (2010). Managing networks through context: Graph visualization and exploration. *Computer Networks, 54*(16), 2809–2824.

Kruja, E., Marks, J., Blair, A., & Waters, R. (2002,). A short note on the history of graph drawing. In *Graph Drawing* (pp. 272–286). Springer Berlin Heidelberg.

Koshman, S. (2004). Comparing usability between a visualization and text-based system for information retrieval. *Journal of Documentation, 60*(5), 565–580.

Lazar, J., Feng, J. H., & Hochheiser, H. (2010). *Research methods in human-computer interaction*. Hoboken, NJ: Wiley.

Mackay, S., Wright, E., Reynders, D., & Park, J. (2004). 3 - EIA-232 overview. In S. Mackay, E. Wright, D. Reynders & J. Park (Eds.), *Practical industrial data networks* (pp. 32–52). Oxford: Newnes.

Mansman, F., Meier, L., & Keim, D. A. (2008). Visualization of host behavior for network security. In J. R. Goodall, G. Conti & K. Ma (Eds.), *Vizsec 2007* (pp. 187-202) Springer Berlin Heidelberg.

Osgood, C. E. (1952). The nature and measurement of meaning. *Psychological Bulletin, 49*(3), 197–237.

Perlman, R. (1985). An Algorithm for Distributed Computation of a Spanning Tree in an Extended LAN. *SIGCOMM Comput. Commun. Rev.*, 15(4), 44–53.

Pietilä, V., Malmberg, T., & Nordenstreng, K. (2004). Teoreettisia yhtymäkohtia ja vastakkainasetteluja - katsaus Suomesta. Teoksessa I. Ruoho, & K. Nordenstreng (toim.), *Tiedotusopin peruskurssin lukemisto*, Tampereen yliopiston Tiedotusopin laitoksen opetusmoniste D46/2004.

Rauterberg, M. (1992). An empirical comparison of menu-selection (CUI) and desktop (GUI) computer programs carried out by beginners and experts. *Behaviour & Information Technology, 11*(4), 227–236.

RFC 958. (1985). *Network Time Protocol (NTP)*. IETF Network Working Group. Available at: http://tools.ietf.org/html/rfc958 (accessed 6 April 2014).

RFC 1034. (1987). *A simple network management protocol*. IETF Network Working Group. Available at: http://tools.ietf.org/html/rfc1034 (accessed 6 April 2014).

RFC 1067. (1988). *A simple network management protocol*. IETF Network Working Group. Available at: http://tools.ietf.org/html/rfc1067 (accessed 6 April 2014).

RFC 1441. (1993). *Introduction to version 2 of the Internet-standard Network Management Framework*. IETF Network Working Group. Available at: http://tools.ietf.org/html/rfc1441 (accessed 6 April 2014).

RFC 2616. (1999). *Hypertext Transfer Protocol – HTTP/1.1*. IETF Network Working Group. Available at: http://tools.ietf.org/html/rfc2616 (accessed 5 April 2014).

RFC 3411. (2002). *An Architecture for Describing Simple Network Management Protocol (SNMP) Management Frameworks*. IETF Network Working Group. Available at: http://tools.ietf.org/html/rfc3411 (accessed 6 April 2014).

RFC 3164. (2001). *The BSD syslog Protocol*. IETF Network Working Group. Available at: http://tools.ietf.org/html/rfc3164 (accessed 6 April 2014).

RFC 3917. (2004). *Requirements for IP Flow Information Export (IPFIX)*. IETF Network Working Group. Available at: http://tools.ietf.org/html/rfc3917 (accessed 6 April 2014).

RFC 5424. (2009). *The syslog protocol.* IETF Network Working Group. Available at: http://tools.ietf.org/html/rfc5424 (accessed 6 April 2014).

RFC 6632. (2012). *An overview of the IETF network management standards.* IETF Network Working Group. Available at: http://tools.ietf.org/html/rfc6632 (accessed 6 April 2014).

RFC 7159. (2014). The JavaScript Object Notation (JSON) Data Interchange Format. Internet Engineering Task Force (IETF). Available at: http://tools.ietf.org/html/rfc7159 (accessed 30 March 2014).

Shiravi, H., Shiravi, A., & Ghorbani, A. (2012). A survey of visualization systems for network security. *IEEE Transactions on Visualization and Computer Graphics, 18*(8), 1313–1329.

Shneiderman, B. (1996). The eyes have it: A task by data type taxonomy for information visualizations. In *Proceedings of the 1996 IEEE Symposium on Visual Languages (VL '96)* (pp. 336–343). Boulder, CO, USA: IEEE.

Spence, R. (2007). *Information visualization: Design for interaction* (2nd ed.). Harlow, England: Pearson Education Ltd.

Staggers, N., & Kobus, D. (2000). Comparing response time, errors, and satisfaction between text-based and graphical user interfaces during nursing order tasks. *Journal of the American Medical Informatics Association, 7*(2), 164–176.

Stallings, W. (1998). SNMP and SNMPv2: The infrastructure for network management. *IEEE Communications Magazine, 36*(3), 37–43.

Stallings, W. (2007). *Data and computer communications* (8th ed.). Upper Saddle River, NJ: Prentice Hall.

Subramanian, M., Gonsalves, T. A., & Rani, N. U. (2010). *Network management: Principles and practice*. Pearson Education India.

Surakka, V., Illi, M., & Isokoski, P. (2004). Gazing and frowning as a new human-computer interaction technique. *ACM Transactions on Applied Perception*, *1*(1), 1–17.

Sähköisen viestinnän tietosuojalaki 16.6.2004/516.

Valtiovarainministeriö. (2010). *Sisäverkko-ohje* (VAHTI 3/2010). Saatavilla: http://www.vm.fi/vm/fi/04_julkaisut_ja_asiakirjat/01_julkaisut/05_valtionhallinnon_tietoturvallisuus/20101203Sisaeve/Sisaeverkko-ohje.pdf (viitattu 7.4.2014).

Vanhala, T. (2005). Kyselylomakkeet käytettävyystutkimuksessa. Teoksessa Ovaska, S., Aula, A. & Majaranta, P. (toim.) *Käytettävyystutkimuksen menetelmät* (Tampereen yliopiston Tietojenkäsittelytieteiden laitoksen raporttisarja B-2005-1). Saatavilla: http://www.cs.uta.fi/usabsem/luvut/2-Vanhala.pdf (viitattu 8.11.2013).

Ware, C. (2013). *Information visualization: Perception for design* (3rd ed.). Boston: Morgan Kaufmann.

Whiteside, J., Jones, S., Levy, P. S., & Wixon, D. (1985). User performance with command, menu, and iconic interfaces. *SIGCHI Bull., 16*(4), 185–191.

Zakas, N. C. (2010). *High performance JavaScript*. Sebastopol, CA: O'Reilly.

Liite 1. Taustatietolomake

Ikä (vuosissa) _____ vuotta

Onko sinulla todettu punavihervärisokeus?
- ☐ kyllä
- ☐ ei

Käytän tekstipohjaista käyttöliittymää
- ☐ jatkuvasti (esim. IRC-keskustelut yms.)
- ☐ useita kertoja päivässä
- ☐ päivittäin
- ☐ 1-2 kertaa viikossa
- ☐ kuukausittain
- ☐ harvemmin

Jos käytät tekstipohjaista käyttöliittymää vähintään viikoittain, kuinka kauan olet käyttänyt tekstipohjaista käyttöliittymää (vuosissa)?
_____ vuotta

Kiitos vastauksistasi!

Testin vetäjä täyttää
Osallistujan numero: _____

Liite 2. Käyttöliittymäesittely

Tekstipohjaisen käyttöliittymän esittely

Kirjaudu kytkimelle

```
$ ssh testi.sw1
```

Näytä kaikkien kytkinporttien tila

```
testi.sw1> sh int status
```

(myös `show interface status`)

Näytä kytkinportin fa0/5 tila

```
testi.sw1> sh int fa0/5 status
```

(tai `sh int status | i fa0/5`)

Poistu kytkimeltä

```
testi.sw1> exit
```

Kytkinportin tila voi olla **"connected"**, **"not connected"** tai **"err-disabled"**

Port	Name	Status	Vlan	Duplex	Speed	Type
Fa0/3		**connected**	1	a-full	a-100	10/100BaseTX

Graafisen käyttöliittymän esittely

Graafisessa käyttöliittymässä on mallinnettu lähiverkon aktiivilaitteita. Navigointi tapahtuu hiirellä osoittamalla. Portit on mallinnettu värikoodein siten, että vihreä vastaa portin tilaa **"connected"**, harmaa tilaa **"not connected"** ja punainen tilaa **"err-disabled"**.

Jos tehtävässä pyydetään tarkistamaan esimerkiksi kytkimen **testi-c.sw3** portin **fa0/27** tila, se tapahtuisi navigoimalla core sw > testi-c.aggr-sw > testi-c.sw3 > testi-c.sw3. Jos kytkinpinossa on vain yksi jäsen, se avataan suoraan. Tarvittaessa saat lisätietoa portista viemällä hiiren osoittimen sen päälle. Löytääksesi oikean portin voit hyödyntää esimerkiksi selaimen find-toimintoa (ctrl-f).

Liite 3. Tehtäväkaavake

Ympäröi mielestäsi oikea vaihtoehto

1. Tarkista kytkimen **talo-b.sw1** portin **fa0/8** tila?	connected	notconnect	err-disabled
2. Tarkista kytkimen **talo-a.sw1** portin **fa0/20** tila?	connected	notconnect	err-disabled
3. Tarkista kytkimen **talo-c.sw1** portin **fa0/24** tila?	connected	notconnect	err-disabled
4. Tarkista kytkimen **talo-a.sw1** portti **fa0/30** tila?	connected	notconnect	err-disabled
5. Tarkista kytkimen **talo-b.sw1** portin **fa0/31** tila?	connected	notconnect	err-disabled
6. Tarkista kytkimen **talo-c.sw1** portin **fa0/1** tila?	connected	notconnect	err-disabled
7. Tarkista kytkimen **talo-a.sw1** portin **fa0/2** tila?	connected	notconnect	err-disabled
8. Tarkista kytkimen **talo-b.sw1** portin **fa0/40** tila?	connected	notconnect	err-disabled
9. Tarkista kytkimen **talo-c.sw1** portin **fa0/30** tila?	connected	notconnect	err-disabled
10. Tarkista kytkimen **talo-a.sw1** portin **fa0/25** tila?	connected	notconnect	err-disabled
11. Tarkista kytkimen **talo-c.sw1** portin **fa0/7** tila?	connected	notconnect	err-disabled
12. Tarkista kytkimen **talo-b.sw1** portin **fa0/17** tila?	connected	notconnect	err-disabled
13. Tarkista kytkimen **talo-c.sw1** portin **fa0/13** tila?	connected	notconnect	err-disabled
14. Tarkista kytkimen **talo-b.sw1** portin **fa0/25** tila?	connected	notconnect	err-disabled
15. Tarkista kytkimen **talo-a.sw1** portin **fa0/40** tila?	connected	notconnect	err-disabled
16. Tarkista kytkimen **talo-c.sw1** portin **fa0/42** tila?	connected	notconnect	err-disabled
17. Tarkista kytkimen **talo-b.sw1** portin **fa0/34** tila?	connected	notconnect	err-disabled
18. Tarkista kytkimen **talo-a.sw1** portin **fa0/14** tila?	connected	notconnect	err-disabled
19. Tarkista kytkimen **talo-c.sw1** portin **fa0/38** tila?	connected	notconnect	err-disabled
20. Tarkista kytkimen **talo-b.sw1** portin **fa0/1** tila?	connected	notconnect	err-disabled
21. Tarkista kytkimen **talo-a.sw1** portin **fa0/7** tila?	connected	notconnect	err-disabled

Testin vetäjä täyttää
Osallistujan numero: _____

Liite 4. Käyttäjätyytyväisyyskysely

Lue huolellisesti jokainen kohta ja ympyröi sopiva numero oman kokemuksesi perusteella jokaiselta asteikolta:

Yleinen arviointi:

Tämä oli mielestäni:

Huono Neutraali Hyvä

-4 -3 -2 -1 0 +1 +2 +3 +4

Miellyttävyyden arviointi:

Käyttäminen tuntui:

Epämiellyttävältä Neutraalilta Miellyttävältä

-4 -3 -2 -1 0 +1 +2 +3 +4

Nopeuden arviointi:

Käyttäminen oli mielestäni:

Hidasta Neutraalia Nopeaa

-4 -3 -2 -1 0 +1 +2 +3 +4

Tehokkuuden arviointi:

Käyttäminen oli mielestäni:

Tehotonta Neutraalia Tehokasta

-4 -3 -2 -1 0 +1 +2 +3 +4

Vaikeuden arviointi:

Käyttäminen oli mielestäni:

Vaikeaa Neutraalia Helppoa

-4 -3 -2 -1 0 +1 +2 +3 +4

Tiedon esitystavan arviointi:

Tämä oli mielestäni:

Sekava Neutraali Selkeä

-4 -3 -2 -1 0 +1 +2 +3 +4

Käyttökelpoisuuden arviointi

Tämä oli mielestäni:

Käyttökelvoton				Neutraali				Käyttökelpoinen
-4	**-3**	**-2**	**-1**	**0**	**+1**	**+2**	**+3**	**+4**

Kokeen vetäjä täyttää:

Osallistujan numero: _____