Nopea tiedonsiirto asennusjärjestelmissä

Pekka Ihalainen

Tampereen yliopisto
Informaatiotieteiden yksikkö
Tietojenkäsittelyoppi
Pro gradu -tutkielma
Ohjaaja: Timo Poranen
Marraskuu 2013

Tampereen yliopisto
Informaatiotieteiden yksikkö
Tietojenkäsittelyoppi
Pekka Ihalainen
Pro gradu -tutkielma, 51 sivua
Marraskuu 2013

Tässä tutkielmassa esitellään, miten ryhmälähetystekniikkaa voidaan käyttää hyväksi suurien datamäärien kopioinnissa sekä käydään lävitse ryhmälähetystekniikkaa soveltavan ohjelmiston suunnittelu, toteutus ja arviointi. Ohjelmistoa arvioidaan kahdella erilaisella testillä, datan siirtonopeustestillä ja ohjelmiston asennusnopeustestillä. Ohjelmiston tuomia etuja tarkastellaan niin yrityksen kuin ohjelmaa käyttävän henkilön näkökulmasta. Huomioitavia laatukriteereitä ovat esimerkiksi varmuus, käytettävyys ja taloudellisuus.

Avainsanat ja -sanonnat: ryhmälähetys, ohjelmistojakelija, arkkitehtuuri

Sisällys

1. Johdanto

Tässä tutkielmassa tarkastellaan, miten ryhmälähetystekniikkaa hyväksikäyttäen suunnitellaan ja toteutetaan työkalu, jonka avulla suurien datamäärien kopiointi useille asiakaspäätteille voidaan suorittaa nopeasti ja vaivattomasti. Tutkielman lähtökohtana on helpottaa ja nopeuttaa tietokoneiden asennusta luokkaympäristössä, jossa koneiden uudelleenasennus on päivittäistä ja koneiden määrä asennusta kohden voi ylittää kolmekymmentä tietokonetta.

Ryhmälähetystä käyttäviä ohjelmia on muutamia, mutta ne suurimmaksi osaksi kopioivat joko kokonaisia levykuvia tai yksittäisiä tiedostoja. Työkalun, joka tässä tutkielmassa suunnitellaan, tulisi olla mahdollista siirtää hakemistorakenteita, joihin sisältyy useita tiedostoja. Työkalu on suunniteltu Tieturi Oy:ssa ja sen toiminnallisuus on optimoitu sopimaan Tieturi Oy:n tarpeisiin.

Seuraavassa luvussa avataan ryhmälähetyksen käyttöä tiedonsiirron välineenä sekä esitellään siihen liittyviä käsitteitä sekä tekniikoita. Kolmannessa luvussa tarkastellaan muutamia jo olemassa olevia ryhmälähetystä käyttäviä sovelluksia sekä esitellään, miten ryhmälähetystä voidaan käyttää hyväksi asennusjärjestelmissä yleisesti. Neljännessä luvussa keskitytään ohjelmistojakelijan suunnitteluun vaatimusmäärittelyn pohjalta sekä esitellään toteutuksessa käytetyt tekniikat kuten esimerkiksi ohjelmointikieli ja apukirjastot. Viidennessä luvussa esitellään ohjelmistojakelijan arkkitehtuuri komponenttitasolla sekä kerrotaan sovelluksen käyttämistä kommunikaatioprotokollista. Kuudennessa luvussa kerrotaan toteutuksen arvioinnista ja sen vahvuuksista ja heikkouksista. Viimeisessä luvussa mietitään kehittämisehdotuksia sekä annetaan yhteenveto ohjelmistojakelijasta.

2. Ryhmälähetystekniikan käyttö tiedonsiirrossa

2.1. Käsitteitä

Tässä luvussa kerrotaan lyhyesti IP-verkon peruskomponenteista sekä IP-verkon rakenteesta. Verkkokomponenteilla tarkoitetaan ohjelmallisia ratkaisuja, joiden avulla verkkoa voidaan käyttää. Kyseiset komponentit on tunnettava, jotta voidaan ymmärtää, miten ryhmälähetys rakentuu ja toimii sekä mitä rajoittavia tekijöitä ryhmälähetyksessä on.

2.1.1. TCP/IP-protokolla

TCP/IP-protokolla (Transmission Control Protocol / Internet Protocol) koostuu useasta Internet-liikennöinnissä käytettävästä tietoverkkoprotokollasta. Vaikka TCP/IP–protokolla ymmärretään usein yhdeksi protokollaksi, on se kahden eri protokollan yhdiste.

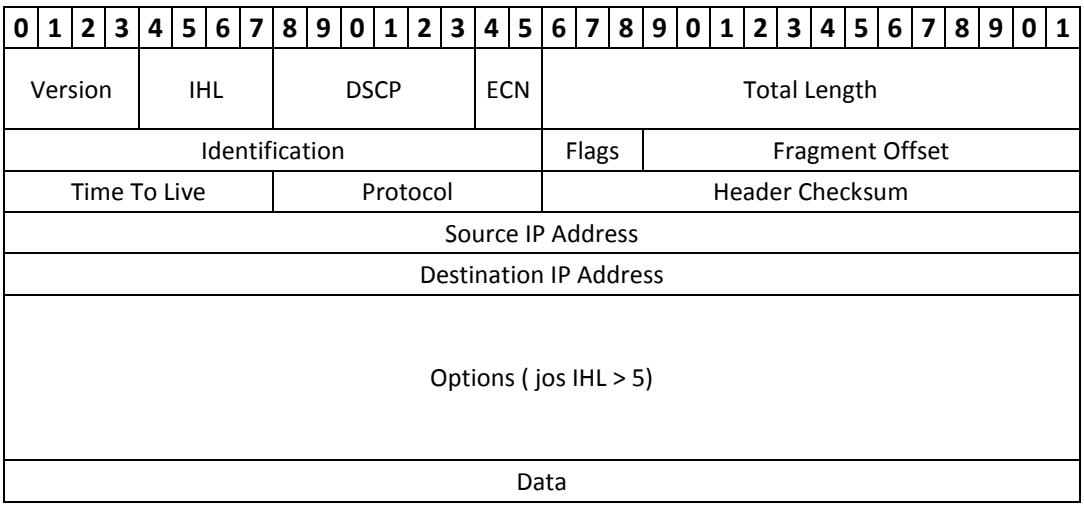
Protokolla määrittelee, miten esimerkiksi verkossa tapahtuva kommunikaatio tapahtuu. Protokolla voidaan havainnollistaa käyttäen vertausta, että protokolla olisi rakennuksen sinikopio. Sen avulla voidaan rakentaa rakennuksen runko ja selvittää, kuinka paljon runko voi pitää sisällään sisältöä, mutta se ei ota kantaa, mitä sisältö on. Jotta protokollaa voidaan käyttää tehokkaasti, on kaikkien kommunikaatioketjun osapuolten, hyväksyttävä käytössä oleva protokolla. Mikäli protokolla on yleisesti hyväksytty, voidaan siitä tehdä tekninen standardi.

IP-protokolla huolehtii, että tietoliikennepaketit reititetään oikeisiin kohteisiin lyhimpiä reittejä käyttäen. IP-protokolla löytää kohteensa IP-osoitteiden perusteella, jotka ovat IP-protokollan versiosta riippuvia kirjain- tai numerosarjoja. Esimerkiksi IPv4:n osoite on numerosarja, joka sisältää neljä kolminumeroista sarjaa pisteellä eroteltuna. Esimerkiksi tietokoneiden käyttämä IP-osoite, jolla kutsutaan itseään, on muotoa ”127.0.0.1”. Tämä IP-osoite on hyväksyttävä osoite, vaikkei siinä ole neljää kolminumeroista sarjaa, koska nollat voidaan unohtaa, mikäli ne ovat päättävän numeron edessä; ”127.0.0.1” on oikeasti muotoa ”127.000.000.001” [IETF, 1981]. Vastaavasti IPv6:n osoite on 128-bittinen osoite, joka jakautuu kahdeksaan 16-bittiseen heksadesimaalisarjaan. Tietokoneiden itseään kutsumisessa käytettävä IPv6 osoite on ”0:0:0:0:0:0:0:1”, joka voidaan myös lyhentää muotoon ”::1”, sillä pelkkiä nollia sisältävät peräkkäiset sarjat voidaan lyhentää kaiksoispisteellä ”::” [Deering & Hinden, 1998].

IPv4 osoitteet ovat jaettu luokkiin (eng. class). Näitä luokkia ovat Luokka A, B, C, D ja E. Luokan A:n osoitteet alkavat ”0.0.0.0”:sta ja päättyvät osoitteeseen ”127.255.255.255”, Luokka B käyttää osoitteita alueelta ”128.0.0.0” – ”191.0.0.0” ja luokka C käyttää osoitteita alueelta ”192.0.0.0” – ”223.0.0.0”. Luokka D on tarkoitettu ryhmälähetyksen käyttöön ja sen osoitealue on ”224.0.0.0” – ”239.255.255.255”. Loput osoitteet kuuluvat luokkaan E, joka on toistaiseksi käyttämätön ja varattu käyttöön tulevaisuudessa. [Microsoft MSDN, 2010]

IPv6-protokolla kehitettiin sen vuoksi, että IPv4-protokollan osoiteavaruus oli rajallinen. Vuonna 2007 RIPE (Network Coordination Center) [RIPE, 2007] arvioi että IPv4:n osoiteavaruus tulee täyttymään vuoteen 2011 mennessä. RIPE:n arvio ei toteutunut tehokkaamman osoitteidenjaon sekä osoitteenmuunnostekniikoiden vuoksi, mutta väistämättä tulee eteen tilanne, jolloin uusia osoitteita ei ole enää antaa. IPv6:ssa mahdollisia osoitteita on 2^121, eli enemmän kuin hiekanjyviä maailmassa.

IP-paketteja voidaan ajaa lähes minkä tahansa verkon päällä, jolloin IP-protokollaa voidaan käyttää yhdistämään eri verkkoja laajemmiksi kokonaisuuksiksi. Esimerkkinä laajasta verkosta on Internet. Verkoissa tietoa välitetään ainoastaan IP-pakettien muodossa reitittimiltä toisille. Reitittimet ovat tietoverkkoja yhdistäviä laitteita, ja ne osaavat lähettää IP-paketin oikeaan suuntaan paketeissa olevien otsikkokenttien (kuva 1) avulla. IP-verkko lukee vain ja ainoastaan IP-paketteja, jolloin verkko ei ole tietoinen mistään ylemmän tason protokollasta, jota IP-paketti voi pitää sisällään. Ylemmän tason protokolla on sisällytetty IP-paketin data-osioon.



Kuva 1: IPv4-paketin rakenne.

Kuvassa 1 on esitetty IPv4-protokollan paketin rakenne. Paketti koostuu otsikkokentästä sekä dataosiosta. Otsikkokentän tehtävänä on määritellä asetuksia sekä kertoa, mistä paketti on tulossa ja mihin paketti on matkalla. IP-paketin versio (version) määritellään ensimmäisissä neljässä bitissä. IPv4:ssa se on aina ’4’ ja vastaavasti IPv6:ssa versio merkitään ’6’. Otsikkotiedot pitävät sisällään version lisäksi myös muita tärkeitä tietoja kuten esimerkiksi paketin elinkaaren (time to live), otsikon tarkistussumman (header checksum) sekä lähdeosoitteen (source IP address) ja kohdeosoitteen (destination IP address). Paketin elinkaaren määrittely tapahtuu käytännössä vähentämällä tässä osiossa olevaa lukua joka kerta, kun paketti saapuu reitittimelle. Kun luku on nolla (0) paketti hylätään ja poistetaan. Näin ollen paketit eivät jää elämään tietoverkoissa vaan

”kuolevat” pois mikäli kohdetta ei tavoiteta. Tarkistussummalla reititin tarkastaa, onko paketti tullut ehjänä perille reitittimeen. Mikäli reitittimen laskema tarkistussumma poikkeaa paketissa olevasta summasta, paketti hylätään.

TCP-protokolla, jonka suunnittelivat Vinton Cerf ja Robert Kahn [1974], on ylemmän tason protokolla, joka sisältyy IP-paketin data-osioon. TCP-protokollan avulla luodaan tietoliikenneyhteyksiä laitteiden välille, jonka jälkeen laitteet pystyvät kommunikoimaan keskenään verkkojen ylitse. Toisin kuin IP-protokolla, jonka tehtävänä on reitittää paketti oikeaan osoitteeseen, TCP-protokollan tehtävä on pitää huolta siitä, että kaikki paketit saapuvat oikeaan päämääräänsä oikeassa järjestyksessä. Mikäli paketteja on hävinnyt tai ne ovat tulleet väärässä järjestyksessä, voidaan hävinneitä paketteja lähettää uudelleen. Tästä syystä TCP-protokollaan on rakennettu erilaisia vuonvalvonta- ja ruuhkanhallintamekanismeja, joiden tarkoituksena on estää pakettien ruuhkautuminen ja sen myötä hukkuminen verkossa.

<table>
<tr><th>0</th><th>1</th><th>2</th><th>3</th><th>4</th><th>5</th><th>6</th><th>7</th><th>8</th><th>9</th><th>0</th><th>1</th><th>2</th><th>3</th><th>4</th><th>5</th><th>6</th><th>7</th><th>8</th><th>9</th><th>0</th><th>1</th><th>2</th><th>3</th><th>4</th><th>5</th><th>6</th><th>7</th><th>8</th><th>9</th><th>0</th><th>1</th></tr>
<tr><td colspan="16">Source port</td><td colspan="16">Destination port</td></tr>
<tr><td colspan="32">Sequence number</td></tr>
<tr><td colspan="32">Acknowledgement number</td></tr>
<tr><td colspan="4">HLEN</td><td colspan="6">Reserved</td><td colspan="6">Flags</td><td colspan="16">Window size</td></tr>
<tr><td colspan="16">TCP Checksum</td><td colspan="16">Urgent pointer</td></tr>
<tr><td colspan="32">Option (if any)</td></tr>
<tr><td colspan="32">Data (if any)</td></tr>
</table>

Kuva 2: TCP-protokollan paketin rakenne.

Kuvassa 2 esitetään TCP-paketin rakenne. TCP-paketin otsikkokenttään kuuluu lähde- ja kohdeportti (source port, destination port), joiden avulla merkitään porttia, jota TCP-protokollan tulisi käyttää. Lisäksi otsikkokenttä pitää sisällään järjestysnumeron (sequence number) ja tunnustusnumeron (acknowledgement number). Niiden avulla voidaan asettaa paketit oikeaan järjestykseen kohteessa sekä pyytää paketteja uudelleen mikäli ne ovat hukkuneet matkalla. Lisäksi paketin oikeellisuus voidaan tarkastaa tarkistussummaa (TCP checksum) hyväksikäyttäen samoin kuin IP-pakettien kohdalla.

Suurin osa verkkoliikenteestä on nykyään TCP-protokollapohjaista. Laajin tällainen verkko on Internet. Esimerkiksi WWW-sivujen hakeminen tapahtuu siten, että verkkoselain ottaa TCP-yhteyden palvelimeen, jonka jälkeen palvelin voi lähettää haluttua dataa TCP-pakettien muodossa takaisin selaimelle. Koska TCP-pakettiliikenne on luotettavaa ja järjestettyä, on luontevaa käyttää

kyseistä protokollaa, kun haluttu tieto on saatava vastaanottajalle kokonaan ilman puuttuvia osia. Esimerkiksi tiedostojen kopioinnin palvelimelta asiakaspäätteelle on oltava järjestäytynyttä sekä luotettavaa, jotta asiakaspääte voi rakentaa identtisen kopion halutusta tiedostosta.

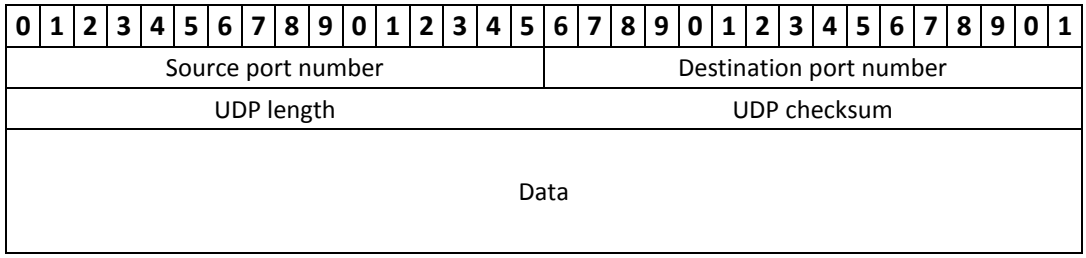
Toisaalta datan ei tarvitse aina tulla varmasti perille tai ei ole väliä, onko se oikeassa järjestyksessä. Esimerkkinä tällaisesta tilanteesta on videoiden katsominen tai äänentoista Internetissä. Videota katsoessa käyttäjä ei huomaa, onko datapaketteja hävinnyt tai ovatko ne väärässä järjestyksessä, sillä kadonneet tai väärässä järjestyksessä tulleet paketit ovat häviävän pieni osa dataa, jota ohjelma näyttää. Tällaisissa tilanteissa voidaan käyttää UDP-protokollaa TCP-protokollan sijasta.

2.1.2. UDP–protokolla

UDP-protokolla (eng. User Datagram Protocol) on ylemmän tason protokolla, jota voidaan käyttää IP-protokollan päällä samaan tapaan kuin TCP-protokollaa. Protokollan suunnitteli David P. Reed vuonna 1980, ja se on virallisesti määritelty RFC 768:ssa. [Postel, 1980]

Toisinkuin TCP-protokolla, jonka tarvitsee suorittaa kättely (eng. handshaking) ennen lähetyksen alkamista sekä yhteyden lopetuksessa, UDP-protokolla käyttää hyvin minimaalista mekanismia aloittaakseen tiedonsiirron kahden tai useamman laitteen välillä. UDP-protokollaa kutsutaankin ”yhteydettömäksi” (eng. connectless) sekä ”tilattomaksi” (eng. stateless) protokollaksi. UDP-protokollan periaate on nopeus, jonka avulla data-paketin vastaanottamis- ja oikeellisuusvarmistukset jätetään sovelluksen huollettavaksi. UDP-protokollaa käytetään useimmiten sovelluksissa, jotka streemaavat videota tai ääntä, ja tietokonepeleissä, joissa vaaditaan reaaliaikaisuutta ja nopeutta verkkoympäristössä. Esimerkkejä sovelluksista, jotka käyttävät UDP-protokollaa, ovat DNS (Domain Name System) [Zytrax, 2013], World of Warcraft (tietokonepeli) [Blizzard, 2012] ja TFTP (Trivial File Transfer Protocol, tiedonsiirtoväline) [RFC 1350].

UDP-protokollan paketti eli datagram on huomattavasti yksinkertaisempi kuin TCP-paketti. Datagram-paketin rakenne (Kuva 3) koostuu lähde- (Source port number) sekä kohdeportin numerosta (destination port number), datan pituudesta (UDP length) ja tarkistussummasta (UDP checksum). IPv4-protokollassa tarkistussumma ja lähdeportin käyttö ovat vapaaehtoisia, toisin kuin IPv6-protokollassa, jossa ainoastaan lähdeportin käyttö on vapaaehtoista. Lähdeportti ja kohdeportti ilmaisevat, mihin porttiin vastaanottaja odottaa UDP-datagrammia ja mihin porttiin vastauksien odotetaan saapuvan. UDP-datagrammin koko on IPv4-protokollassa 65507 bittiä, kun taas IPv6-protokollassa datagrammin koko voi maksimissaan olla 2^31-1 bittiä, eli 4294967287.



Kuva 3: UDP-datagrammin rakenne.

UDP-protokollaa voidaan käyttää erityisesti tiedonsiirron välineenä nopeutensa vuoksi. Tämä kuitenkin edellyttää, että sovellustasolla pakettien oikea järjestys tarkastetaan ja hukkuneet paketit lähetetään uudelleen. Esimerkkinä UDP:n päälle rakennetuista protokollista, jotka toteuttavat näitä edellytyksiä, ovat RUDP (Reliable User Datagram Protocol) sekä DCCP (Datagram Congetsion Control Protocol). Kun UDP-protokollaa käyttävään tiedonsiirtosovellukseen lisätään mahdollisuus käyttää ryhmälähetystä (eng. multicasting) saadaan data siirtymään yhdestä koneesta usealle laitteelle erittäin nopeasti.

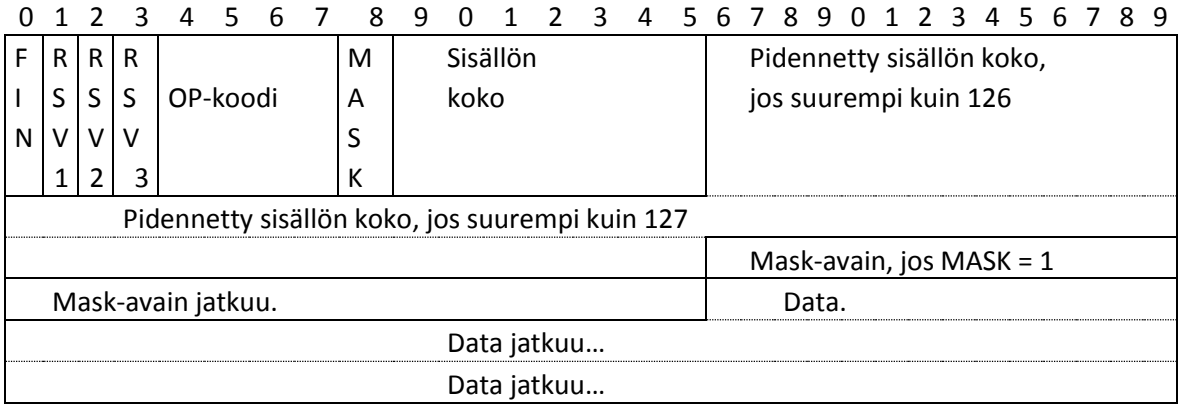
2.1.3. Websocket-protokolla

WebSocket on vuonna 2011 IETF:n (Internet Engineering Task Force) standardoima protokolla [RFC 6455, 2011]. WebSocket API on W3C:n (World Wide Web Consertium) standardoima. WebSocketin avulla voidaan luoda TCP/IP-yhteys minkä tahansa sovelluksen ja palvelimen välille, myös web-sivun ja palvelimen välille. WebSocketin selainkäyttö on rajattu uusimpiin selainsovelluksiin (Opera 11, Safari 5, Safari mobile iOS4.2, Firefox 6 ja Internet Explorer 10), mutta vanhemmissakin selaimissa on joko tuki aiempiin WebSocket-versioihin tai mahdollisuus asentaa kolmannen osapuolen laajennus, joka mahdollistaa WebSocketin käytön.

WebSocket-paketin rakenne on kuvan 4 mukainen. Yläreunassa kulkevat numerot merkitsevät bittejä, joiden mukaan nähdään, kuinka monta bittiä kukin osio käyttää paketin koosta. Ensimmäinen osa paketista on yhden bitin mittainen FIN (eng. final) osio, joka tarkoittaa sitä, onko kyseinen paketti viestin viimeinen. Mikäli FIN on asetettu bitiksi 1, tulkitaan se viestin viimeiseksi paketiksi. On tärkeää huomata, että numero ’1’ ei tarkoita ascii-merkkiä ’1’ vaan binäärikoodia ’1’. RSV1-3 -kentät ovat lisälaajennuksien merkitsemistä varten. Yleisesti RSV1-3 -kentät ovat asetettu binäärikoodiksi 0, mutta mikäli WebSocket:lle on sovelluksen kehittäjän toimesta rakennettu esimerkiksi oma laajennus, voidaan kyseisen laajennuksen käyttö ilmoittaa näissä kolmessa kentässä. OP-koodi ilmaisee vastaanottajalle, minkälainen paketti on kyseessä. OP-koodien avulla voidaan kertoa, onko paketti tarkoitettu WebSocketin sisäiseen sykkeen testaukseen (PING- ja PONG-paketit) vai onko paketti yhteyden lopettamispyyntö ja onko paketti esitetty teksti- vai binäärimuodossa. MASK ilmoittaa, onko kyseisen paketin data suojattu avaimella. Mikäli se on suojattu, avain löytyy Mask-avain kohdasta. Kaikki paketit asiakassovellukselta palvelimelle pitää

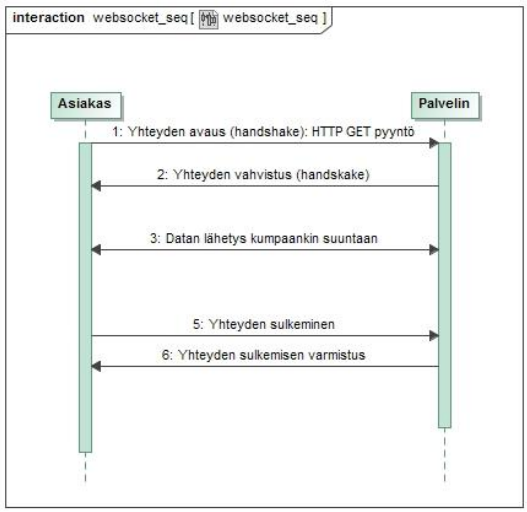
olla suojattuja ja kaikki paketit palvelimelta asiakassovellukselle eivät saa olla suojattuja. Mikäli näin ei ole, palvelimen ja asiakassovelluksen on suljettava avoin yhteys välittömästi.

Sisällön koko määrittää, kuinka paljon dataa paketin dataosio pitää sisällään. Sisällön koko ilmoitetaan seitsemänä (7) bittinä, 7+16 bittinä tai 7+64 bittinä. Mikäli paketti on lähetetty asiakassovelluksesta palvelimelle, määritellään seuraavaksi ’Mask-avain’, joka on 32 bitin pituinen avain, jolla palvelin pystyy avaamaan lähetetyn paketin datan. Mask-avaimen jälkeen tulee paketin dataosio, joka pitää sisällään lähetettävän viestin joko teksti- tai binäärimuodossa. Dataosion muoto on ilmoitettu OP-koodilla.



Kuva 4: WebSocket-paketin rakenne.

WebSocket-protokollan toiminta on kuvattu sekvenssikaaviossa kuvassa 5. Protokollassa yhteys palvelimen ja asiakkaan välillä pitää sisällään kaksi vaihetta: kättelyvaiheen sekä datansiirtovaiheen. Kättelyvaihe pitää sisällään yhteyden avaamisen palvelimen kanssa (kuva 5, kohta 1), jossa asiakas lähettää palvelimelle http GET -pyyntönä halun nostaa protokollataso http-protokollasta WebSocket-protokollaan. Mikäli asiakkaan kättely on oikeellinen, palvelin vastaa pyyntöön ja yhteys avautuu asiakkaan kanssa (kuva 5, kohta 2). Kättelyn jälkeen alkaa datansiirto, jonka aikana niin asiakas kuin palvelin voivat samanaikaisesti lähettää viestejä toisilleen (kuva 5, kohta 3). Toisin kuin tavallisessa http-palvelinyhteydessä, yhteys WebSocket-protokollassa on kokoajan aktiivisena. Yhteys suljetaan palvelimen ja asiakkaan välillä, kun jompikumpi osapuoli lähettää paketin, jonka OP-koodina on yhteyden katkaisupyyntö (kuva 5, kohdat 5 ja 6).



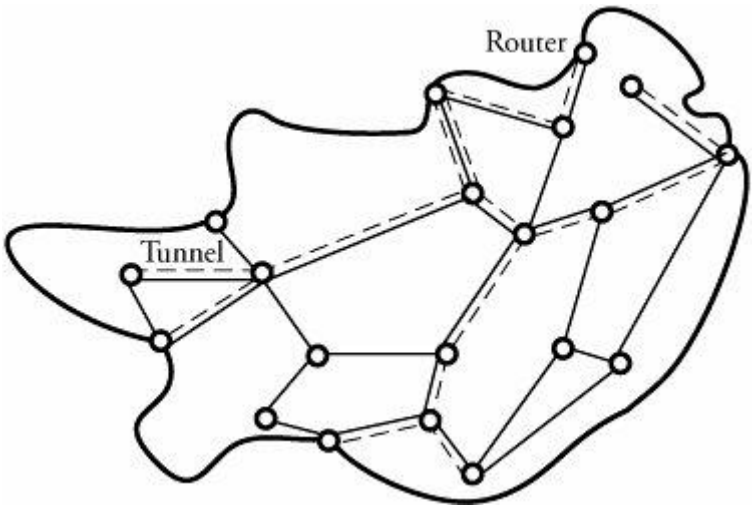
Kuva 5: WebSocket protokollan toiminta.

2.2. Ryhmälähetys

Ryhmälähetys on tekniikka, jolla voidaan lähettää dataa IP-verkossa yhdeltä monelle (eng. one-to-many) tai monelta monelle (eng. many-to-many) reaaliajassa. Sille on tunnusomaista, että lähettäjän ei tarvitse tuntea vastaanottajia tai niiden määrää. Ryhmälähetystekniikan suunnitteli Steeve Deering vuonna 1986 [StarDush, 2000]. Hän sai ryhmälähetyksen ja IPv6:n parissa tehdystä työstään IEEE:n Internet Award -palkinnon vuonna 2010 [IEEE, 2012]. Ryhmälähetys oli ensimmäistä kertaa laajassa Internet-kokeilussa vuonna 1992, kun Internet Engineering Task Force (IETF) kokeili audiodatan lähettämistä Internetissä ryhmälähetyksen avulla San Diegossa. Kaksi vuotta myöhemmin Rock-yhtye Rolling Stones esiintyi Dallasissa, jossa konserttia saattoi kuunnella Internetin kautta eripuolilla maailmaa. Rolling Stonesin konsertti oli ensimmäinen suuri ”kyberavaruuskonsertti” ja ensimmäinen ”rasitustesti” MBONE (eng. multicast backbone) järjestelmälle. [Savetz et al., 1996]

MBONE on kokeilullinen selkäranka - infrastruktuuri - IP-ryhmälähetykselle Internetissä. Sen suunnittelivat Van Jacobson, Steve Deering sekä Stephen Casner vuonna 1992 [Lewis, 1995]. MBONE:n tarkoitus on vähentää liikkuvan datan määrää audio- ja video konferensseissa, joissa osallistujia on eripuolilla maailmaa. MBONE on maksuton sekä kaikkien käytettävissä, sillä samankaltaisen palvelun kaupallistaminen on hankalaa. Suurimmassa osassa Internetissä olevissa reitittimissä ryhmälähetys on pois päältä, koska ryhmälähetysdatan virtaa on vaikea hallita, eikä käyttöoikeuksia pystytä rajaamaan. Lisäksi Internetin palveluntarjoajille ryhmälähetysliikennemaksun laskeminen on hankalaa.

MBONE on virtuaalinen verkko, joka koostuu useasta ryhmälähetystä tukevasta verkosta Internetin päällä. MBONE käyttää hyväkseen tunnelointiprotokollaa, jonka avulla ryhmälähetyspaketit voidaan lähettää tavallisen IP-paketin sisällä, tällöin paketit saavuttavat määränpäänsä, vaikka matkalla olisikin ryhmälähetystä tukemattomia reitittimiä. MBONE käyttää vain pientä osaa (224.2.0.0) class D IP-osoiteavaruudesta (224.0.0.0 – 239.255.255.255), joka on määritelty ryhmälähetyksille.



Kuva 6: MBONE virtuaaliverkon topologia.

Kuvassa 6 on esitetty, miten MBONE virtuaaliverkko on toteutettu kuvitteellisessa verkkoympäristössä. Ympyrät kuvaavat reitittimiä, jotka lähettävät paketteja toisilleen. Jotkin reitittimet ovat yhdistetty toisiinsa käyttäen katkoviivaa. Ne kuvaavat tunnelointia kahden reitittimen välillä. Tunneloinnin avulla mahdollistetaan ryhmälähetyksessä käytettyjen pakettien siirtäminen reitittimeltä toiselle IP-pakettien muodossa ilman, että reitittimen tarvitsee tukea ryhmälähetystä. Nykyisin MBONE on käytössä useimmissa korkeakouluissa, yliopistoissa sekä tutkimuslaitoksissa. MBONE virtuaaliverkkoa käytetään nykyisin nimenomaan konferenssien pitämiseen sekä jaettuihin työpöytiin.

2.2.1. Reititysmallit

Reititysmalleja, joilla paketit kulkevat verkossa lähettäjältä vastaanottajalle tai useille vastaanottajille, on viisi kappaletta. Kukin reititysmalleista toimii hieman eri tavoin verrattuna toisiinsa ja kullakin on omat käyttötarkoituksensa sekä vaatimuksensa. Reititysmalleja tarvitaan, jotta voidaan kontrolloida, miten tieto liikkuu verkossa ja kenelle. Tiedon vastaanottaja ja lähettäjä eivät ole verkossa koskaan vierekkäin, jolloin tiedon on kuljettava reitittimien kautta. Jotta tieto kulkisi lyhyintä ja nopeinta reittiä lähettäjältä vastaanottajalle, on käytettävä algoritmeja laskemaan optimaalinen reititys.

Käytettävissä olevia reititysmalleja ovat täsmälähetys (eng. unicast), jokulähetys (eng. anycast), yleislähetys (eng. broadcast), ryhmälähetys (eng. multicast) sekä maalähetys (eng. geocast). Vaikka reititysmallit eivät kaikki ole yhteydessä ryhmälähetykseen, on alla kuvattu lyhyesti kunkin reititysmallin perusperiaate ja käyttötarkoitus.

Täsmälähetys on reititysmalli, jossa lähettäjä lähettää tietoa vain yhdelle vastaanottajalle, käyttäen yksilöllistä IP-osoitetta. Täsmälähetystä käytetään, kun asiakas pyytää yksilöllistä tietoa lähettäjältä. Esimerkki tällaisesta tiedosta on Internetissä toimivat www-sivut, joissa selain pyytää www-palvelimelta sivua http-kutsulla ja palvelin lähettää pyydetyn sivun datan selaimelle. Täsmälähetys on eniten käytetty reititysmuoto Internetissä juuri http-liikenteen vuoksi. Täsmälähetystä käytetään myös sovellusten välisessä kommunikoinnissa sekä tiedonsiirrossa; esimerkiksi tiedostojen kopioinnissa palvelimelta asiakaspäätteelle. Koska täsmälähetys on yksi-yhdelle (eng. one-to-one), sen soveltuvuus esimerkiksi nettiradion tai videopalvelujen reititysmalliksi ei ole hyvä siksi, että se vie huomattavan paljon verkkokaistaa verrattuna esimerkiksi ryhmälähetykseen.

Toisin kuin täsmälähetys jokulähetys (eng. anycast) toimii periaatteella, että paketti lähetetään IP-osoiteryhmälle, jonka jälkeen paketti reititetään lähimmälle vastaanottajalle kyseisessä ryhmässä. Ryhmän jäsenet voivat olla eripuolilla verkkoa eikä lähettäjä tiedä, mikä jäsenistä on lähinnä. Jokulähetyksen käyttämä Border Gateway -protokolla (BGP) valitsee lähimmän vastaanottajan automaattisesti. Jokulähetystä käytetään palveluissa, joissa vaaditaan nopeaa reaktioaikaa sekä korkeaa toimintavarmuutta. Esimerkiksi DNS-palvelut (eng. Domain Name System), jotka ovat vastuussa IP-osoitteiden muuttamisesta paremmin ymmärrettäviksi nimiksi, käyttävät jokulähetystä. DNS-palvelimia voi olla yhdessä ryhmässä useita, jolloin nimikutsut reitittyvät aina lähimpään ja helpoiten tavoitettavaan palvelimeen riippuen asiakkaan sijainnista verkossa. Jokulähetyksen ansiosta nimikyselyt toimivat vaikka osa DNS-palvelimista ei olisikaan saavutettavissa. Tuolloin nimikysely ohjautuu seuraavaksi lähimpään palveluun.

Yleislähetys, toisin kuin täsmä- ja jokulähetykset, on osoitettu kaikille vastaanottajille tietyssä verkossa. Yleislähetyksen periaate on, että data lähetetään tiettyyn verkkosegmenttiin, jolloin jokainen pääte kyseisessä segmentissä voi vastaanottaa lähetetyn datan. Verkkosegmenttejä voidaan rajata käyttäen esimerkiksi reitittimiä tai virtuaalisia lähiverkkoja. Yleislähetys on tarkoitettu lähinnä suljettuihin verkkoihin, kuten LAN-verkkoteknologioihin. Esimerkiksi Internet ei tue yleislähetystä kuten ei myöskään X.25 [Cisco, 2012]. Syynä tähän on, että yleislähetystä käytettäessä jokainen verkkosegmentissä saa lähetetyn datan riippumatta siitä, tahtoiko kyseinen asiakas sitä vai ei. Tämä kuormittaa verkon kapasiteetin äärimmilleen, minkä seurauksena muut verkkosegmentissä olevat palvelut kärsivät.

Ryhmälähetyksen periaate on sama kuin yleislähetyksenkin. Suurimpana erona näiden kahden välillä on, miten tieto vastaanotetaan sekä lähetetään. Ryhmälähetyksessä lähetetty tieto ei ”vuoda” verkkosegmenttiin, vaan tiedosta kiinnostuneiden vastaanottajien pitää ilmaista kiinnostuksensa tiettyä ryhmälähetysryhmää kohtaan.

Ryhmät ovat IPv4-osoiteavaruudessa osoitteesta 224.0.0.0 osoitteeseen 239.255.255.255. Kyseistä ryhmitystä kutsuttiin aiemmin Class D osoiteryhmäksi. Tästä arkkitehtuurista luovuttiin, sillä IP-osoitteiden määrä kasvoi räjähdysmäisesti 1990-luvun lopulla. Nykyisin käytössä on CIDR (Classless Inter-Domain Routing), joka on metodi jonka päämääränä oli hidastaa IP-osoitteiden loppumista. Koska ryhmälähetystä käyttävä lähettäjä lähettää vain dataa niille verkkoasiakkaille, jotka ovat ilmaisseet kiinnostuksen kyseiseen dataan, ei verkon kuormitusta tapahdu samassa mittakaavassa, kuin yleislähetystä käytettäessä.

Ryhmälähetyksessä vastaanottajien lukumäärää ei ole rajoitettu, ja teoriassa lähetettyä tietoa voi vastaanottaa lukematon määrä verkkoasiakkaita ilman, että datan siirtonopeus kärsii. Todellisuudessa nopeus kuitenkin on määriteltävä hitaimman verkkoasiakkaan mukaan, sillä mikäli nopeus valittaisiin sattumanvaraisesti tai nopeimman verkkoasiakkaan mukaan, hitaammat verkkoasiakkaat joutuvat tilanteeseen, jossa tietoa tulee liian nopeasti ja ne eivät pysy mukana. Verkkoasiakkaiden nopeuden erot syntyvät verkkokorttien erilaisuudesta, kiintolevyjen kirjoitusnopeudesta sekä verkossa liikkuvasta muusta datasta.

Yleisimmin käytetty protokolla ryhmälähetystä käytettäessä on UDP. Koska UDP on järjestäytymätön ja siinä ei ole sisäänrakennettua mekanismia, jolla voitaisiin tarkistaa, onko tieto saapunut perille ja oliko se oikeassa järjestyksessä, käytetään UDP:n päällä sovellustason lisäprotokollia, jotka toteuttavat kyseisen mekanismin. Nämä protokollat pitävät huolen, että jokainen lähetystä kuunteleva verkkoasiakas pysyy mukana. Siirtonopeutta voidaan muuttaa ja lähetys voidaan jopa pysäyttää hetkeksi, mikäli hitaimmat asiakkaat tarvitsevat uudelleenlähetystä tai aikaa prosessoida jo lähetettyä tietoa. Esimerkiksi RUDP (Reliable User Datagram Protocol) on yksi käytetyistä protokollista ryhmälähetyksen kanssa.

Maalähetys ei varsinaisesti ole standardoitu reititysmalli mutta tunnettu käsite liittyen reititysmalleihin. Maalähetyksellä tarkoitetaan useimmiten ryhmälähetystä, joka tapahtuu maanosien välillä. Maalähetyksen avulla voidaan kohdistaa maanosittain esimerkiksi mainontaa, kommunikointia tai rajata palveluiden saantia. Yksi suurista käytännön sovellutuksista on matkapuhelinten verkkomahdollisuuksien rajaaminen. Maalähetyksen avulla voidaan esimerkiksi älypuhelimia käskeä olla ottamatta yhteyttä verkkoon jonkin tietyn maan alueella tai rajoittaa joitakin puhelimen verkko-ominaisuuksia, kuten GPRS-yhteyksiä [RFC 2009].

2.2.2. Sovellusalueet

Ryhmälähetys toteutettiin saman aikaan kuin TCP-protokolla otettiin käyttöön, eli vuonna 1988. Tuolloin sen käyttö jäi kuitenkin hyvin vähäiseksi. Syynä tähän oli, että ei ollut vertaisverkkoja, Internetiä tai sovelluksia, jotka olisivat voineet käyttää hyväkseen ryhmälähetystä. Ryhmälähetystekniikan käyttöä ei myöskään edistänyt se, että suurin osa teleoperaattoreista ei hyväksynyt UDP:stä muodostettua liikennöintiä. Osaltaan käyttöä myös viivästytti mahdolliset hyväksikäytöt puutteellisen tietoturvan takia kyseisessä tekniikassa. Jokainen saattoi ilmaista halukkuutensa ottaa vastaan tietoa, vaikkei heillä olisi ollut tarpeeksi nopeaa yhteyttä ja siten hidastivat tai pysäyttivät muidenkin vastaanottajien tiedonsiirron (DDoS). Voidaan sanoa, että ryhmälähetystekniikka keksittiin liian aikaisin. Koska käyttöä tälle tekniikalle ei ollut, se ”unohdettiin”, kunnes Internet yleistyi 1990-luvulla.

Internetin yleistymisen jälkeen ryhmälähetykselle on auennut uusia mahdollisuuksia, kuten esimerkiksi digitaaliset radio- tai videolähetykset, tiedon kopiointi, pelit sekä IRC. IRC on verkkopalvelu, joka mahdollistaa reaaliaikaisen keskustelun sekä tiedonsiirron Internetissä useiden käyttäjien välillä.

Radio- sekä videolähetyksiä voidaan ryhmälähetyksen avulla tuottaa suurelle yleisölle ilman suurta verkkokaistan kulutusta. Radio- ja videolähetyksissä ei ole myöskään tarpeen käyttää sovellustason protokollia, jotka pitäisivät huolta datapakettien perillepääsystä tai oikeasta järjestyksestä, sillä datapaketit ”vanhentuvat” nopeasti, eikä hukkapaketeilla ei ole suurta merkitystä lopputuloksen kannalta. Esimerkiksi videota katsellessa voi kuva joskus pysähtyä tai vääristyä hetkeksi mikä tarkoittaa, että osa datapaketeista ei ole tullut perille tai paketit ovat saapuneet virheellisessä järjestyksestä. Tästä huolimatta uusien pakettien saapuessa video jatkuu normaalisti.

Nykyään suurin osa peleistä käyttää Internettiä, jotta pelaajat voivat pitää yhteyttä tai pelata toistensa kanssa. Jotta pelit toimisivat nopeasti sekä saumattomasti, on yhteydenpidon asiakassovelluksien välillä oltava nopeaa ja reaaliaikaista. Ryhmälähetystä voidaan käyttää mielekkäästi siinä osassa kommunikaatiota, jossa tieto halutaan siirtää jokaiselle tai suurelle osalle asiakassovelluksia nopeasti. Peleissä sekä sovelluksissa on tosin muistettava, että tiedon oikeellisuus on tärkeää, joten siitä huolta pitävä protokolla on toteutettava ryhmälähetyksen päälle. Esimerkiksi IRC (Internet Relay Chat), joka on verkossa toimiva keskustelupalvelu, käyttää ryhmälähetystä tiedon lähettämiseen yhdeltä käyttäjältä toisille käyttäjille. IRC-arkkitehtuuri rakentuu siten, että käyttäjät ovat yhteydessä palvelimiin, jotka ovat yhteydessä toisiin palvelimiin eli käytössä on ns. ”spanning tree”-arkkitehtuuri. Jokainen palvelin toimii keskussolmuna muulle sen näkemälle verkolle. Asiakas siis ei ole tietoinen muista käyttäjistä vaan palvelimet pitävät kirjaa asiakaspäätteistä ja välittävät asiakaspäätteiltä tulleita viestejä eteenpäin muille.

Tiedonsiirrossa ryhmälähetystä on käytetty jo kauan; esimerkiksi kun kovalevystä tehdään varmuuskopio (eng. backup) tai kun kovalevystä otetaan levykuva (eng. disc image) ja siirretään se muille koneille. Levykuvaa käytetään, kun vaikkapa jonkin koulun tai yrityksen koneet halutaan asentaa kaikki samanlaisella konfiguraatiolla ilman, että jokainen kone asennetaan erikseen. Tällöin riittää, kun yksi kone asennetaan ja kovalevyn sisältö sekä rakenne siirretään sellaisenaan toisiin koneisiin. Lopputulos on, että koneet ovat ohjelmiltaan ja asetuksiltaan identtiset alkuperäisen koneen kanssa ja asennuksiin käytetty aika on murto-osa siitä, mitä olisi kulunut jos jokainen kone oltaisiin asennettu manuaalisesti. Tiedonsiirrossa ei voi käyttää ryhmälähetystä natiivisti nykyisissä käyttöjärjestelmissä. Mikäli ryhmälähetystä tahdotaan käyttää täytyy käyttöjärjestelmiin kopioida kolmannen osapuolen ohjelmia, joiden avulla ryhmälähetystekniikka saadaan käyttöön.

Esimerkkejä tämänkaltaisista ohjelmista ovat Symantec Ghost (erikoistunu levykuvan siirtoon), Microsoft System Centre Configuration Manager, SCCM, (erikoistunut ohjelmien asennukseen sekä levykuvan siirtoon) sekä UDPCast (vain levykuvan siirto). Vaikka ryhmälähetys tekniikkana on vanha ja ollut jo kauan käytössä, tiedonsiirrossa sitä on käytetty hyvin vähän, mikäli ei lasketa mukaan levykuvien siirtoa tai varmuuskopiointia. Esimerkiksi vasta SCCM:n versio 2008 otti käyttöön ryhmälähetyksen asennustiedostojen jaossa, vaikka SCCM oli ollut käytössä jo vuodesta 1994. Symantec Ghost ja UDPCast lukevat kovalevyä bittitasolla eivätkä tiedostotasolla, mikä tarkoittaa etteivät ne ole kykeneviä siirtämään kuin yhden tiedoston kerrallaan. Tästä syystä ne eivät sovellu monen tiedoston tai hakemistorakenteen siirtämiseen (esimerkiksi asennusmedia) palvelimelta asiakaspäätteille.

2.2.3. Käytön haasteet

UDP-protokolla tuo omat haasteensa sovellustason toteutukselle, kun käytetään ryhmälähetystä tiedonsiirrossa. UDP:n tilattomuus ja järjestäytymättömyys sovellusta suunniteltaessa on otettava huomioon. Suunniteltaessa UDP:n käyttöä on tarkasteltava, tarvitaanko kyseisessä sovelluksessa tiedon saapumista oikeassa järjestyksessä ja ilman datapakettien katoamista. Mikäli kyse on tiedostojen kopioinnista, on tarkoituksenmukaista, että tiedonsiirto tapahtuu UDP:n päällä toimivalla ryhmälähetyksellä samankaltaisesti kuin jos käytettävä alemman tason protokolla olisi IP. Sovellustasolla pitää myös huolehtia siitä, että kaikki asiakkaat pysyvät samassa tahdissa muitten kanssa. Käytännössä tämä toteutetaan yleensä sisäisellä käskynjakokanavalla, joka toimii sovelluksen ja palvelimen välisenä yksityisenä TCP/IP-sokettiyhteytenä (one-to-one) (eng. socket). Käskynjakokanavan kautta asiakaspäätteet ilmoittavat palvelinsovellukselle, mikä on viimeisin niiden saama paketti, tai onko asiakassovelluksella liian monta pakettia käsittelemättä. Näissä tilanteissa palvelin reagoi lähettämällä vaaditut paketit uudelleen tai pysäyttämällä koko tiedonsiirron kunnes asiakassovellus on valmis vastaanottamaan uusia paketteja.

Verkkotasolla ryhmälähetys asettaa tiettyjä vaatimuksia, jotta se toimisi moitteettomasti sekä tehokkaasti. Esimerkiksi yrityksessä tai akateemisessa laitoksessa on yleensä käytössä sisäverkossa monia aliverkkoja, joilla saadaan eriytettyä sekä rajattua eri osia sisäverkosta. Tällöin ryhmälähetystä käytettäessä reitittimien tulee olla ryhmälähetystä tukevia sekä kyllin nopeita. Mikäli reitittimet eivät tue ryhmälähetystä, eivät ne pysty kloonaamaan palvelimelta lähetettyjä paketteja kaikille osallistujille. Jos reitittimet ovat liian hitaita verrattuna muuhun ympäröivään verkkoon, voivat ne ylikuormittua, jolloin ne pudottavat paketteja pois ja tiedonsiirto katkeaa sovelluksen toimesta. Ideaalitilanne on, että tiedonsiirron nopeus rajoittuu, ei niinkään verkoon, vaan asiakaspäätteen kovalevyjen kirjoitusnopeuteen tai palvelimen kovalevyn lukunopeuteen.

2.3. Ryhmälähetyksessä käytetyt protokollat

IGMP (Internet Group Management Protocol) on kommunikaatioprotokolla, jota käytetään paikallisten reitittimien ja asiakaspäätteiden välissä muodostettaessa ryhmälähetysryhmää. IGMP on integroitu osa ryhmälähetystä ja se toimii IPv4-verkossa. IPv6-verkossa IGMP:tä ei tueta vaan ryhmälähetys hallinnan on korvannut Multicast Listener Discovery (MLD). IGMP:stä on yhteensä kolme standardia määritelty IETF:n (Internet Engineering Task Force) RFC (Request for Comments) dokumentteihin. IGMPv3 [RFC 4604, 2006] on viimeisin versio, joka päivittää niin IGMPv1:stä [RFC 1112, 1989] kuin myös IMGPv2:sta [RFC 2236, 1997]. IGMPv3 päivittää edeltäviä standardeja lisäämällä mahdollisuuden ryhmälähetysisäntien poistua ryhmälähetysryhmästä sekä mahdollisuuden rajata ryhmälähetysryhmän muodostamista vain tietyistä IP-osoitteista.

Procol Independent Multicast (PIM) on protokolla, jonka avulla luodaan datanvälitys yhdeltä-monelle tai yhdeltä-yhdelle. PIM ei itsessään sisällä mekanismia, joka toimisi verkkotopologian löytäjänä, vaan käyttää siinä apunaan perinteisiä protokollia. PIM jakautuu neljään eri variaatioon: harva tila (eng. sparse mode) [RFC 4601, 2006], tiheä tila (eng. dense mode) [RFC 3973, 2005], kaksisuuntainen PIM (eng. bidirectional mode) [RFC 5015, 2007] sekä lähdespesifi PIM (eng. source-specific mode) [RFC 3569, 2003].

Harva tila on PIM:n eniten käytetty muoto, sillä sen skaalautuvuus suurelle käyttäjämäärälle on parempi kuin esimerkiksi tiheän tilan. Harvassa tilassa PIM rakentaa puun johon valitaan ryhmälähetysryhmään haluavien asiakkaiden jokin paikallinen reititin vastaamaan kyseisen aliverkon liikenteestä. Valittua reititin välittää viestit asiakkailta puun juurelle, joka on ennalta määrätty RP-reititin (Rendezvous Point). Näin ollen saadaan jaettu puu (eng. RP Tree), joka toimii karttana ryhmälähetyksessä.

Tiheä tila, kaksisuuntainen PIM sekä lähdespesifi PIM ovat harvoin käytettyjä muotoja ja siksi ne on tässä tutkielmassa jätetty pois tarkemmasta esittelystä.

3. Ryhmälähetys käytännössä

Tässä luvussa esitellään kaksi esimerkkiä olemassa olevista ohjelmista, jotka toteuttavat tiedonsiirrossa ryhmälähetystä ja yleislähetystä. Ohjelmat ovat olleet jo kauan saatavilla, joten niissä ei ole havaittavissa enää yleisiä virheellisiä toimintoja ja ne ovat hyvin suosittuja ja arvostettuja käyttäjien keskuudessa. Esimerkit ovat valittu kirjoittajan työkokemuksen pohjalta, joten kyseiset ohjelmat eivät ole ainoita eivätkä välttämättä käytetyimpiä ryhmälähetysohjelmia.

3.1. Asennusjärjestelmistä

Asennusjärjestelmät ovat yhdestä tai useammasta palvelimesta ja sovelluksesta koostuva kokonaisuus, jonka tarkoituksena on helpottaa ja nopeuttaa tietokoneiden sekä muiden IT-laitteiden ylläpitoa ja asennusta. Varhaisimmat asennusjärjestelmät ovat ajalta, jolloin tietokoneet vasta tekivät tuloaan yleiseen tietouteen. Esimerkiksi Microsoftin System Management Server (SMS) – nykyään System Center Configuration Manager (SCCM) – tuotiin markkinoille ensimmäistä kertaa vuonna 1994 ja on ollut siitä lähtien saatavilla [Wikipedia, 2012]. Muita vastaavia järjestelmiä ovat mm. Symantec GhostCast Suite sekä Symantec Altiris.

Asennusjärjestelmän käyttöönoton myötä on yrityksissä tai muissa organisaatioissa mahdollisuus saavuttaa tuntuvia parannuksia tiedonhallinnassa ja IT-kustannuksissa. Niiden laajuus tosin riippuu henkilö- sekä laitemäärästä kyseisessä yrityksessä. Mikäli yrityksessä on, esimerkiksi kymmenen erillistä laitetta, on hyöty miltei olematon. Tämä johtuu siitä, että asennusjärjestelmien käyttöönotto sekä ylläpito ja hallinta vaativat lisätyötä. Tällöin työ, joka menee asennusjärjestelmän ylläpitoon voitaisiin suunnata vaihtoehtoisesti suoraan huolettaviin laitteisiin. Mikäli yritys on keskisuuri tai suuri, on asennusjärjestelmän tuoma hyöty suurempi. Esimerkiksi siinä ajassa, jossa yhden IT-laitteen asennus toteutetaan manuaalisesti, voidaan asennusjärjestelmän avulla asentaa miltei mikä tahansa lukumäärä laitteita. [Microsoft, 2012]

Muita hyötyjä asennusjärjestelmän käyttöönotossa on, että yritys voi ulkoistaa tietohallinnon kolmannelle osapuolelle, jolloin yrityksen henkilökunta voi keskittyä omaan osaamisalueeseen sekä kohdentaa voimavaransa paremmin. Asennusjärjestelmän avulla voidaan myös päätelaitteiden hallinta keskittää, jolloin ylläpidettävyys parantuu ja työmäärä laitetta kohden pienenee. Muita ilmeisen selviä hyötyjä ovat tietoturvan mukaantulo keskitetyssä ylläpidossa sekä graafinen hallinta asiakaspäätteille ja palvelimille.

Tässä tutkielmassa asennusjärjestelmistä tutkitaan vain ohjelmien asennukseen liittyviä piirteitä, vaikka asennusjärjestelmät yleisellä tasolla pitävät sisällään myös tietoturvaylläpidon, tietokantojen

hallinnan sekä mobiililaitteiden hallinnan. Pääpaino asennuksiin liittyvissä seikoissa on asennuksen nopeudessa ja vaivattomuudessa.

3.2. Olemassaolevia sovelluksia

3.2.1. Microsoft System Center Configuration Manager

System Center Configuration Manager (SCCM) [Microsoft, 2013] on Microsoftin tuottama järjestelmänhallintasovellus, jonka tarkoituksena on helpottaa ja nopeuttaa Windows-pohjaisten päätelaitteiden ja mobiililaitteiden asennusta, ylläpitoa ja hallintaa. SCCM on saanut alkunsa vuonna 1994, kun ensimmäinen Systems Management Server (SMS) julkaistiin. SMS nimi kulki mukana vuoteen 2007 kunnes Microsoft uudisti sovellusta ja muutti nimen System Center Configuration Manageriksi.

SCCM on suunniteltu ensisijaisesti suurille yrityksille, joissa IT-hallinnon piiriin kuuluu suuri määrä tietokoneita. SCCM:n käyttöönotto vaatii jo valmista IT-infrastruktuuria toimiakseen. Active Directory -palvelu sekä SQL-palvelin ovat oltava asennettuina ennen kuin SCCM pystyy toimimaan tehokkaasti. Active Directory (AD) on Windows-palvelimen ominaisuus, jonka avulla asiakaspäätteitä voidaan liittää toimialueeseen ja sitä kautta yhdistetyn hallinnon piiriin. Active Directory itsessään vaatii myös DHCP- sekä DNS-palvelimien olemassaolon ja oikeellisen konfiguroinnin Active Directoryä varten. SCCM:n vahvuus tulee esille siinä, että SCCM hyväksikäyttää Active Directory ja siinä olevia tietoja asiakkaana olevista tietokoneista. Tämän lisäksi SCCM:ään voidaan integroida Microsoftin tietoturvapalvelu WSUS (Windows Server Update Services), jolloin SCCM:n kautta voidaan hallinnoida asiakaspääteiden Windows- ja tietoturvapäivityksiä.

System Center Configuration Managerin suurin etu, edellä mainituista huolimatta, on mahdollisuus etäasentaa asiakaspäätteitä ajastetusti joko loppukäyttäjälle sopivana ajankohtana tai pakotetusti järjestelmänvalvojan päättämänä hetkenä. SCCM pystyy asentamaan miltei mitä tahansa ohjelmia, kunhan asiakaspäätteet täyttävät ohjelman itsensä määrittelemät vaatimukset. Joissakin tapauksissa ohjelmiin joudutaan tekemään asennussarjoja (eng. task sequence), joidenka avulla SCCM ohjaa asennuksen etenemistä. Asennusmedia, jota SCCM käyttää etäasennuksessa, on SCCM:n omilla jakopalvelimilla (eng. distribution points). Asiakaspäätteet voivat suoraan asentaa ohjelman näiltä palvelimilta tai ensin kopioida asennusmedia paikalliselle kovalevylle ja aloittaa asennuksen sieltä.

SCCM 2007 R2 -versiossa tuli mahdolliseksi käyttää ryhmälähetystä niin levykuvien kuin myös ohjelmien jakamisessa jakopalvelimilta asiakaspäätteille. Tämä on huomattava etu ympäristöissä, joissa on suuri määrä koneita ylläpidossa tai koneita, joita asennetaan usein uudelleen esimerkiksi luokkakoneet. SCCM:n heikkous kuitenkin on se, että asennusjärjestelmää ei pysty pakottaa aloittamaan asennusta juuri sillä hetkellä, kun asentaja niin tahtoo. Koska SCCM on suunniteltu

suurille ja globaaleille ympäristöille, on siinä viiveitä asennuksien aloittamisissa. Syynä näihin on SCCM:n käyttämän agentin ja palvelimen keskustelusykli. Agentin avulla SCCM saa tietoonsa asennetut ohjelmat sekä laitekokoonpanon asiakaspäätteeltä. Agentti myös ottaa vastaan SCCM:n lähettämät asennus-, päivitys- ja hallinnointikäskyt. SCCM:n ja asiakaspäätteen agentin keskustelusykli voidaan määritellä, mutta mitä lyhempi sykli on, sitä enemmän ylimääräistä liikennettä verkossa kulkee. Kokemuksen kautta on todettu myös, vaikka sykli on määritelty lyhyeksi (1 minuutti), asennus ei välttämättä silti lähde käyntiin, vaan odottaa mahdollisesti tuntejakin. Suuressa yrityksessä tämän kaltainen odottelu ei haittaa, sillä asennukset tehdään yleensä yöllä, jolloin asennettavalla koneella ei ole muuta käyttöä. Luokkaympäristöissä asennus on saatava lähtemään heti ja asennus on varmistettava toimivaksi jälkikäteen, eikä odottelu ole suotavaa.

3.2.2. Symantec Ghost Solution Suite

Ghost Solution Suite (GSS) [Symantec, 2013] on Symantecin kehittämä asennusjärjestelmä, jonka avulla voidaan hallita ja asentaa asiakaspäätteitä. GSS tukee niin yksittäisten ohjelmien asennusta tehtävien kautta (eng. tasks) kuin myös kokonaisten kovalevyjen kloonaamista yhdestä koneesta toisiin.

Ensimmäinen versio Symantec Ghost sovelluksesta julkaistiin vuonna 1997. Symantec Ghostia käytetään kovalevyjen kloonaamisessa. Sittemmin sen ympärille on rakennettu kokonainen asennusjärjestelmä hyväksikäyttäen sovelluksia, jotka Symantec on ostanut ja integroinut omiin tuotteisiinsa. Ensimmäinen Symantec Ghost Suite julkaistiin vuonna 2004, jolloin Symantec muutti Ghostin Enterprise version nimen Ghost Suiteksi ja näin tehtiin suurempi ero yritysversion ja kuluttajaversion välille.

Ghost Suiten avulla on mahdollista hallita asennettuja ohjelmia asiakaspäätteillä sekä asentaa niitä yhdelle tai useammalle asiakkaalle samanaikaisesti. GSS on erikoistunut levykuvien sekä ohjelmien asentamiseen sekä varmuuskopiointiin. Sen sijaan SCCM on erikoistunut enemmän hallintaan ja ylläpitoon, vaikkakin Ghost Suite tarjoaa myös tuen asentaa tietoturvapäivityksiä sekä pitää yllä inventaariota asiakaspäätteistä. Ghost suite voidaan integroida Active Directoryyn siten, että AD:sta saadaan asiakaskoneiden tilit suoraan Ghost Suiten käyttöön. AD:n kautta voidaan myös asentaa Ghost Agent -ohjelma asiakaspäätteille suoraan. Ghost Agentia tarvitaan yhteydenpitoon Ghost Suiten käyttämän palvelimen kanssa.

Ghost Suitessa ohjelmien asennus toimii samankaltaisesti SCCM:n kanssa asennussarjoja käyttäen. Ghost Suitessa voidaan valita ohjelmistoja asennettaessa käyetäänkö tiedonsiirrossa palvelimelta asiakaspäätteille ryhmälähetystä vai täsmälähetystä.

Toisin kuin SCCM:ssä Ghost Suiten asennukset alkavat samalla hetkellä kun asennuskäsky palvelimelta on annettu, mikä tuo GSS:n lähemmäksi haluttua asennusjärjestelmää luokka-asennuksien näkökulmasta. Silti tässäkin tapauksessa asennuksiin käytettävä aika kasvaa GSS käyttäessä verrattuna puoliautomaattiseen asennusjärjestelmään. Puoliautomaattisessa asennusjärjestelmässä käytetään pohjana yhtä Windows-käyttöjärjestelmästä otettua levykuvaa, jonka päälle asennetaan halutut ohjelmat sekä vaadittavat tiedostot manuaalisesti. Kun pohjakuva on valmis, siitä otetaan Ghostilla uusi levykuva, joka kopioidaan toisille tietokoneille. Mikäli luokka-asennukset tehtäisiin käyttäen pelkästään GSS:ää, jouduttaisiin jokaiselle asennettavalla ohjelmalle tekemään oma asennussarja. Koska koulutukset voivat olla hyvin erilaisia, on ohjelmia sekä niiden versioita lukematon määrä. Tällöin asennussarjojen ylläpitämiseen ja tekemiseen kuluu huomattavasti enemmän aikaa kuin manuaaliseen asennukseen.

3.2.3. UDPCast

UDPCast on Linux-pohjainen työkalu, josta on olemassa myös Windows-versio. UDPCastilla voidaan lähettää dataa samanaikaisesti usealle eri tietokoneelle ryhmälähetystekniikalla. UDPCastin peruskäyttötarkoitus on lähettää halutun kovalevyn tai osion data toiseen kovalevyyn, jolloin niistä tulee identtisiä kopioita. Tällöin yhden kokonaan asennetun tietokoneen kopiointi toisiin tietokoneisiin voidaan suorittaa helposti käyttäen tätä ohjelmaa.

UDPCastille voidaan tehdä asetustiedosto, jonka mukaan ohjelma valitsee halutut ajurit, verkkoasetukset, kovalevyn ja pakkausalgoritmin. Sovellus voidaan käynnistää joko suoraan käyttöjärjestelmästä, USB-muistitikulta, CDROM-levyltä tai verkon kautta käyttäen PXE-Boot mahdollisuutta. UDPCastin vahvuuksiin kuuluu mahdollisuus konfiguroida se halutulla tavalla sekä mahdollisuus käynnistää se useasta eri lähteestä. Myös mahdollisuus valita kahdesta eri pakkausalgoritmista (GZIP, LZOP) on erinomainen, sillä pakkaus vaikuttaa siihen kuinka paljon verkko kuormittuu kopioinnin yhteydessä.

Vaikka UDPCastin konfigurointi voidaan lukea sen vahvuudeksi, on siinä havaittavissa myös heikkouksia. Esimerkiksi jokainen ajuri pitää lisätä suoraan ohjelman asennusmediaan, eikä siinä ole käyttäjäystävällistä käyttöliittymää helpottamaan tehtävää. Ajurien asennusta ja hankintaa vaikeuttaa myös se, että UDPCast on Linux-pohjainen työkalu, jolloin on mahdollista, että sopivia ajureita kaikille laitteille ei ole saatavilla, mikäli laitevalmistajat eivät tue Linux käyttöjärjestelmää.

Koska UDPCast kopioi joko partition tai koko kovalevyn toisiin tietokoneisiin, on myös pidettävä huolta siitä, että kovalevyt on samanlaisia tai vähintäänkin samankokoisia. Mikäli konekanta ei ole standardoitua, on suuri mahdollisuus, että UDPCast ei toimi jokaisessa tietokoneessa tai kopiointi keskeytyy, mikäli kovalevyt ovat eri kokoisia kapasiteetiltaan.

3.2.4. Symantec Ghost

Ghost-kloonaussovellus [Symantec Ghost, 2013] on alun perin suunnitellut Murray Haszard vuonna 1995. Hän toimi yrityksessä nimeltä Binary Research, jolta Symantec osti Ghost-sovelluksen vuonna 1998. Vuonna 2003 Symantec osti sen hetkisen suurimman kilpailijansa PowerQuest-sovelluksen ja liitti sen osaksi Ghost-sovellusta, mikä johti uuden sukupolven Ghost version julkaisuun. Kehitystyö on jatkunut tähän päivään saakka ja siihen on lisätty uusia ominaisuuksia, kuten esimerkiksi Hyper-V -tuki sekä mahdollisuus kloonata ja palauttaa VMWare-virtuaaliympäristöjä.

Ghost on osa laajempaa Symantec Ghost Solution Suite -”perhettä”. Ghost Solution Suite on asennusjärjestelmä, jonka avulla voidaan asentaa ja hallinnoida koneita etäältä. Ghost on Solution Suitin komponentti, jonka tehtävänä on vain kopioida tai siirtää levykuvia koneilta toisille.

Ghost-sovellusta voidaan käyttää joko itsenäisesti tai yhteydessä palvelimeen. Itsenäisesti käytettynä sovelluksella on mahdollista tehdä varmuuskopio tietokoneesta tai palauttaa levykuva tietokoneeseen, missä sovellus on käytössä sillä hetkellä (LiveImaging). Palvelimeen yhdistettynä tietokoneesta saadaan levykuva palvelimelle, joka voidaan palauttaa verkon ylitse useille koneille. Ghost-sovellusta käytettäessä on mahdollisuus hyödyntää kolmea eri ryhmälähetystekniikkaa: täsmälähetystä, ryhmälähetystä ja jokulähetystä. Kun asennetusta tietokoneeta otetaan levykuva palvelimelle, käyttää Ghost aina täsmälähetystä. Riippuen tilanteesta, levykuvan palautus voidaan toteuttaa kaikilla kolmella menetelmällä.

Toisin kuin UDPCastissa, Ghost tukee Windowsin 32bit- sekä 64bit-ympäristöjä sekä vanhempaa DOS-ympäristöä. Tästä syystä Ghost voidaan lisätä esimerkiksi WinPE (Windows Preinstallation enviroment) ympäristöön, jolloin Ghost saadaan toimimaan helposti PXE-ympäristön jatkeena. Koska WinPE on minimalistinen Windows-käyttöjärjestelmä, voidaan siihen asentaa kaikki saatavilla olevat Windows-ajurit. Suurin osa laiteajureista on saatavilla Windows-ympäristöön suoraan WinPEhen asennettuna tai suoraan kopioitavana laiteiden valmistajilta. Tästä syystä WinPEn ja Ghostin yhdistelmä toimii suurimmassa osassa ympäristöjä, vaikka laitekanta olisikin hajanainen.

Levykuvan teon yhteydessä voidaan valita millä teholla Ghost pakkaa datan levykuvaan. Riippuen pakkauksen tehosta levykuvan koko voi pienentyä jopa kuuteenkymmeneen prosenttiin alkuperäisestä, mutta viedä enemmän aikaa, kun tarkastellaan yhteensä levykuvan tekoon ja palautukseen käytettyä aikaa (taulukko 1).

Pakkaustaso	Kuvan koko (megabittiä)	Varmuuskopio (Disk-To-Image)		Palautus (Image-To-Disk)		Pakkaussuhde
		Aika	Aika prosentuaalisesti pohjasta	Aika	Aika prosentuaalisesti pohjasta	
Ei pakkausta (pohja)	1,065	5 min. 27 sek.	100	9 min. 4 sek.	100	100
Kevyt (alhainen)	725	5 min. 4 sek.	93	7 min. 29 sek.	82	68
Keskiverto	620	7 min. 57 sek.	131	8 min. 6 sek.	89	58
*Raskas	600	12 min. 49 sek.	158	7 min. 59 sek.	88	56

Taulukko 1: Vertailu eri pakkaustasojen suhteesta toisiinsa [Symantec, 2009].

Symantec [2009] on vertaillut eri pakkaustasojen suhdetta ja palautus- sekä varmuuskopiointinopeuksia. Taulukossa 1 esitetään tuloksia testistä, jossa varmuuskopioitiin neljän gigatavun kovalevy (SCSI IBM DDRS-34560DD). Levyn sisältönä oli 1065 megatavua satunnaista dataa. Varmuuskopioitu kone oli malliltaan Dell Optiplex GX1 P3 450. Varmuuskopio palautettiin saman koneen toiselle kovalevylle (IDE Samsung VA34323A), joka oli kooltaan yhtä suuri. Pakkaustasot olivat ei pakkausta, kevyt, keskiverto sekä raskas. Mikäli pakkaustasoksi valittiin ’Raskas’, oli mahdollista valita lisäargumentiksi vielä lisäasetus, jonka avulla voitiin lisäsäädellä ’Raskas’-pakkaustason nopeutta ja pakkaussuhdetta. Taulukossa 1 tähti tarkoittaa, että ’Raskas’-pakkaustaso käytti suurinta mahdollista pakkaussuhdetta ’Z9’ (Z voi olla väliltä 1-9), jolloin varmuuskopiointi vei eniten prosessoritehoja ja aikaa pakkausta tehdessä. [Symantec, 2009]

Taulukosta 1 huomataan, että käyttämällä *Raskas-pakkausta levykuvan koko pienenee parhaimmillaan 40%, mutta pakkaukseen käytetty aika kasvaa huomattavasti. Tämän vuoksi kokonaisaika levykuvan tekemiseen ja palauttamiseen on suurempi kuin jos pakkausta ei käytettäisi laisinkaan. Sen sijaan testin mukaan kevyellä pakkauksella suoritettu levykuvan varmuuskopiointi ja palautus on kaikista nopein vaihtoehto. Kevyessä pakkauksessa pakkaussuhde on hyvä, joten levykuvan koko pienenee ja pakkaukseen käytetty aika voitetaan lähettämällä vähemmän dataa verkon ylitse.

4. Ohjelmistojakelijan suunnittelu

Tässä luvussa esitellään ohjelmistojakelijan suunnittelussa huomioitavia seikkoja, kuten esimerkiksi Tieturin infrastruktuuri sekä datan lähetykseen, vastaanottamiseen ja pakkaukseen sisältyviä tekijöitä. Suunniteltaessa on otettava huomioon myös ohjelmointikielet sekä apukirjastot, joiden avulla saavutetaan haluttu toiminnallisuus ohjelmaan. Lopuksi esitellään lyhyesti ohjelmistojakelijan pääkomponentit ja niiden haluttu toiminnallisuus. Tämän luvun tarkoitus on antaa yleiskuva ohjelmistonjakelijan toiminnallisuudesta sekä niistä vaatimuksista, joita sille on asetettu. Seuraavassa luvussa perehdytään syvällisemmin ohjelmiston arkkitehtuurin sekä tapaan, jolla ohjelmisto toteutetaan.

4.1. Tiedonsiirron vaatimukset Tieturi Oy:ssä

Tieturi on projektinjohtamiseen ja tietotekniikkakoulutukseen ja valmennukseen erikoistunut yritys, jonka pääpaikka on Ruotsissa. Suomessa yritys toimii Helsingissä ja Tampereella. Tampereella koulutustiloja on kolme, joissa on yhteensä 36 tietokonetta. Koulutuksien aihe voi eri tiloissa vaihtua päivittäin, jolloin myös tietokoneiden asennukset vaihtuvat koulutukseen sopiviksi. Tämä asettaa erityisiä haasteita koneiden asennuksille, koska tietokoneiden tulee olla räätälöity juuri kyseiseen koulutukseen sopiviksi.

Tietokoneet voidaan asentaa joko manuaalisesti siten, että jokainen kone asennetaan erikseen tai käyttäen ryhmälähetystä, jolloin riittää yhden koneen esiasennus, joka kopioidaan muihin tietokoneisiin. Ryhmälähetys on suurimmassa osassa asennuksia nopein vaihtoehto, vaikka tällöinkin jotkin ohjelmat pitää asentaa manuaalisesti esiasennuksen jälkeen jokaiselle koneelle erikseen. Tämä esiasennuksen jälkeinen manuaalinen ohjelmien asennus on kaikkein aikaa vievin vaihe koneiden asennuskaaressa.

Esimerkkinä voidaan käyttää Office 2010 -versiota, jota ei voida asentaa suoraan esiasennukseen lisenssiehtojen vuoksi, jolloin se täytyy asentaa jokaiselle koneelle erikseen. Office 2010 -asennusmedia on kooltaan noin 4 GT (4000 MT) ja asennuspalvelimelta asennettuna yksi asennus vie nykyaikaisella tietokoneella noin kaksikymmentä minuuttia. Mikäli Office halutaan asentaa jokaiselle koulutustilan koneelle, joudutaan tekemään jopa kuusitoista eri asennusta. Tällöin asennusaika kasvaa huomattavasti, koska asennusmedian kopiointi verkkolevyltä on sitä hitaampi, mitä enemmän kopioivia asiakaspäätteitä palvelimella on. Voidaan arvioida karkeasti, että mikäli verkon nopeus on 100 Mt sekunnissa, neljä gigatavua voi kopioitua palvelimelta asiakkaalle noin kuudessa minuutissa. Tästä seuraa, että mikäli asiakkaita on 16, niin kopiointiaika on 111 minuuttia eli miltei kaksi tuntia. Tähän on syynä se, että jokaisen asiakkaan voidaan olettaa käyttävän saman verran verkkokaistaa kuin muutkin. Tällöin 100 megatavun verkossa jokainen asiakas saa 6 megatavun kaistan itselleen eli kopiointinopeudeksi tulee 600 Kt sekunnissa.

Edellä olevan esimerkin lisäksi jokaiselle koneelle erikseen kopioitavia tiedostoja tai sovelluksia voi olla useita. Tämän takia olisi mielekästä, että asennusmedioita ja erillisiä tiedostoja sekä hakemistorakenteita voitaisiin esiasennuksen jälkeenkin kopioida ryhmälähetystä käyttäen jokaiselle koneelle nopeasti ja vaivattomasti. Tällöin asennusmedian kopiointinopeus jokaiselle koneelle lähentelisi verkon teoreettista maksiminopeutta eli samaa nopeutta, kuin jos yksi asiakas kopioisi tiedostoja palvelimelta.

Pahimmassa tilanteessa tutkielman esimerkkinä olevan Tieturin Tampereen koulutustilojen asennukseen voi mennä aikaa useita tunteja luokkaa kohden. Mikäli asennusaika saataisiin esimerkiksi ryhmälähetystä käyttäen puolitettua verrattuna aikaan, joka kuluu ”ylimääräisten” ohjelmien asentamiseen, olisi se suuri etu niin asentajalle ajallisesti kuin myös yritykselle rahallisesti.

4.2. Vaatimusmäärittely

Toteutettavan ohjelmistojakelijan vaatimukset voidaan jakaa toiminnallisiin ja ei-toiminnallisiin vaatimuksiin. Toiminnalliset vaatimukset määrittelevät yleisesti, mitä järjestelmän tai ohjelman odotetaan tekevän. Esimerkkinä toiminnallisista vaatimuksista voi olla, miten ohjelma kommunikoi ympäristönsä kanssa tai miten käyttäjät työskentelevät ohjelman kanssa. Ei-toiminnalliset vaatimukset määrittelevät reunaehtoja ohjelmalle tai järjestelmälle. Ei-toiminnallisia vaatimuksia ovat esimerkiksi vastausajat, käytettävyys, laite- tai ohjelmistorajoitukset. Ohjelmistojakelija on pääasiallisesti suunniteltu käytettäväksi asennuksien helpottamiseen Tieturi Oy:ssä, jolloin vaatimusmäärittely on yksinomaan rakennettu vaatimuksista ja rajoitteista, joita Tieturi luo ohjelmalle sekä siitä, mitä ohjelmaa käyttävät tukihenkilöt ovat toivoneet siltä. Tästä syystä osa arkkitehtuuripäätöksistä ei välttämättä toimi muissa ympäristöissä. Samoin joitain toiminnallisuuksia, joita voidaan odottaa tämän kaltaiselta ohjelmalta, on jätetty lisäämättä. Esimerkki rajoittavista ei toiminnallisista ratkaisuista on päätös, että ohjelman ei tarvitse tukea Linux-käyttöjärjestelmää, koska Tieturi Oy toimii pääasiallisesti Windows-ympäristössä.

4.2.1. Toiminnalliset vaatimukset

Vaatimusmäärittelyssä tärkeimmiksi toiminnallisiksi vaatimuksiksi nousivat *käytettävyys* [ISO 9241-11, 1998], *luotettavuus* [IEEE Reliability Society, 2013] sekä *saatavuus* [Federal Standard 1037C, 1996]. Käytettävyydellä vaatimusmäärittelyssä tarkoitettiin käyttöliittymää niin palvelin- kuin asiakasohjelmassa ja sen helppokäyttöisyyttä ja nopeutta. Käyttöliittymän tulisi olla selkeä, eikä siinä tulisi olla enempää valintoja tai mahdollisuuksia kuin yleensä on tarpeellista. Mikäli joitain lisävalintoja tarvitaan, ne voidaan suorittaa komentoriviltä käyttöliittymää käynnistäessä. Palvelimen käyttöliittymässä tulisi olla mahdollisuus valita kopioitava tiedosto tai hakemistorakenne sekä mahdollisuus vaihtaa kanavaa, johon asiakassovellus ottaa yhteyttä. Palvelinsovelluksen tulisi myös siirtoa tehdessä näyttää, kuinka kauan siirtoa on toteutettu ja kuinka

kauan sitä on vielä jäljellä sekä prosentuaalinen osuus siirretystä datasta verrattuna koko datan määrään. Käytettävyyden osalta asiakassovelluksessa on tärkeätä, että sovellus on käynnistystilassa valmis ottamaan yhteyden palvelimeen ja aloittamaan datan vastaanottamisen ilman asetuksia muuttamatta. Yhteydenottokanava sekä datan säilytyskohde tulisi olla vakioita, mutta muutettavissa tarvittaessa. Tämä nopeuttaa datan siirron aloittamista ja helpottaa käyttäjien työmäärää siinä ettei käyttäjien tarvitse asettaa jokaisen asiakassovelluksen asetuksia uudelleen joka asennuskerralla. Käyttöliittymän olisi hyvä olla minimalistinen ja piilossa, mutta avattavissa mikäli asetuksia tahdotaan muuttaa.

Luotettavuudella vaatimusmäärittelyssä tarkoitetaan sitä, että ohjelman tulisi pystyä raportoimaan, onko tiedonsiirto onnistunut vai ei. Lisäksi ohjelman tulisi pystyä antamaan raportti siitä, miksi tiedonsiirto on epäonnistunut, mikäli niin on käynyt. Ryhmälähetystä käyttäessä on odotettavissa, että jokainen paketti ei tule perille asiakassovellukselle. Siksi on tärkeätä pystyä myös todentamaan reaaliaikaisesti, että jokainen asiakas on samassa tahdissa toistensa kanssa. Palvelimen tulee siis pystyä kontrolloimaan lähetettyä tietoa lähetyksen aikana ja tarvittaessa sen on pystyttävä lähettämään tieto uudelleen asiakkaille, jotka sitä tarvitsevat. Asiakassovelluksen odotetaan olevan käynnissä koko ajan ja käynnistyvän automaattisesti koneen käynnistyessä. Lisäksi asiakassovelluksen tulee ottaa yhteys datanjakelukanavan avulla palvelimeen sekä varmistaa, että yhteys pysyy päällä ja yrittää yhteydenottoa uudelleen tietyn väliajoin mikäli yhteyttä ei ole voitu muodostaa. Jatkuva yhteys palvelinsovellukseen varmistaa, että kopioinnin alustus tapahtuu nopeasti ilman, että käyttäjien tarvitsee aloittaa erikseen kopiointia jokaisella asiakaspäätteellä.

Saatavuudella ohjelmistojakelijan osalta tarkoitetaan mahdollisuutta käyttää palvelimen sovellusta monesta eri lähteestä, esimerkiksi älypuhelimesta, www-sivulta tai suoraan palvelimelta. Tällöin on tärkeätä, että asentajat voivat aloittaa kopioinnin, vaikka he eivät olisikaan fyysisesti tietokoneen ääressä ja yhteydessä palvelimeen. Jotta käyttöliittymä olisi mahdollisimman saavutettavissa sekä toimisi usealla eri laitteella, on käyttöliittymä valittu näytettäväksi www-sivuna. Käyttöliittymää voi siis ohjata miltä tahansa www-selainta ajavalta laitteelta, kuten esimerkiksi Tablet PC:ltä, älypuhelimesta tai PC:ltä. Näin ollen asentaja voi esimerkiksi tehdä asiakaspäätteillä muita asennustöitä ja aloittaa uuden asennusmedian kopioimisen lopettamatta sen hetkisiä töitä. Tämä antaa myös mahdollisuuden etäasentaa asiakaspäätteitä VPN-yhteyden avulla ilman, että tarvitsee olla fyysisesti paikanpäällä. VPN-yhteys muodostaa salatun yhteyden kahden tietokoneen, esimerkiksi kotitietokoneen ja työpaikan palvelimen välillä, jolloin yhteyden ottanut tietokone pääsee käsiksi yrityksen sisäiseen verkkoon.

4.2.2. Ei-toiminnalliset vaatimukset

Vaatimusmäärittelyssä määriteltiin ei-toiminnallisiksi vaatimuksiksi Tieturin sisäisen verkon rajoitukset sekä nopeuteen ja tiedonsiirtotapaan vaikuttavat seikat. Suurimmaksi sekä myös

ohjelmalle tärkeimmäksi tiedonsiirtoon liittyväksi seikaksi voidaan nostaa vaatimus, että ohjelmistojakelijan tulee käyttää ryhmälähetystekniikkaa, mikäli tiedon vastaanottajia on useampi kuin yksi (Liite 1, kohta 1). Mikäli vastaanottajia on vain yksi, voidaan lähetysmuotona käyttää joko täsmälähetystä tai ryhmälähetystä. Tiedonsiirron nopeuden kannalta täsmälähetys on hiukan nopeampi yhden vastaanottajan tilanteessa, sillä silloin on mahdollista käyttää TCP/IP-protokollaa UDP-protokollan sijasta. TCP/IP-protokollaa käytettäessä säästytään ylimääräisiltä sovelluksen tarkistuksilta silti osin, onko tieto päässyt perille ja onko se järjestäytynyttä.

Ryhmälähetyksen käytön lisäksi vaatimusmäärittelystä käy ilmi, että sovelluksen tulee käyttää sisäiseen käskynjakokanavaan TCP/IP-protokollaa (Liite 1, kohta 2). Käskynjakokanavan kautta asiakassovellukset ovat jatkuvasti yhteydessä palvelinsovellukseen. Käskynjakokanavan avulla asiakassovellukset saavat tärkeät tiedotteet ja käskyt palvelimelta, kuten milloin tulee olla valmis vastaanottamaan datalähetystä tai mitä pakkausalgoritmia on käytetty lähetetyn datan pakkauksessa. Palvelinsovelluksen kuin myös asiakassovelluksen tulee pystyä pakkaamaan ja purkamaan käyttäen hyväksi GZIP-algoritmia. Pakkaussuhdetta pitää pystyä myös muuttamaan käyttäen GZIP-algoritmin ’Z’-apumäärettä (Liite 1, kohta 3). ’Z’-apumääreen avulla voidaan algoritmia käskeä parantamaan pakkaussuhdetta pakkausnopeuden kustannuksella.

Toinen ohjelmistojakelijan tärkeistä piirteistä on, että sillä tulee pystyä lähettämään hakemistorakenteita tiedostoineen eikä pelkästään yksittäisiä tiedostoja (Liite 1, kohta 4). Tämän piirteen avulla esimerkiksi laajat sovelluksien asennusmediat on mielekästä kopioida nopeasti palvelimelta usealle koneelle käyttäen ohjelmistojakelijaa tavallisen kopioinnin sijaan. Jotta saavutettaisiin mahdollisimman nopea siirto eli käytetään mahdollisimman vähän aikaa itse verkon kuormittamiseen, pakataan hakemistorakenne ennen itse tiedonsiirron aloittamista. Näin ollen lähetettävä data on pakatussa muodossa ennen lähetystä eikä tiedonsiirrossa käytetä ylimääräistä aikaa ja prosessointitehoa pakkaukseen ja purkamiseen. Datan purkaminen tapahtuu jokaisella asiakaspäätteellä tiedonsiirron loputtua.

4.3. Datan pakkaus

Vaatimusmäärittelyssä esitettyä GZIP-pakkausalgoritmia käytetään ohjelmistojakelijan pakkauksessa. GZIP on hyvin tuettu valmiissa ohjelmointikirjastoissa usealla ohjelmointikielellä. Esimerkiksi ’zlib’-kirjaston lisenssi antaa täydet valtuudet ohjelmoijalle käyttää kyseistä kirjastoa omiin tarkoituksiinsa, olivatpa ne kaupalliset tai eivät. GZIP-ohjelman ovat kehittäneet Jean-Loup Gailly ja Mark Adler [GZIP.org, 2003], ja siitä tuli jakeluun ensimmäinen käytettävä versio vuonna 1992. GZIP:n alkuperäinen tarkoitus oli korvata Unix-käyttöjärjestelmissä ollut ’compress’, koska algoritmit, joita ’compress’ sekä useat muut silloiset pakkausohjelmat käyttivät, olivat patenttisuojattuja. Tämä haittasi näiden pakkausohjelmien laajaa käyttöä. GZIP sekä sen käyttämä algoritmi suunniteltiin GNU-projektille [GNU.org, 2013], jotta patenttikysymyksistä päästäisiin

eroon. GZIP käyttää algoritmissään pohjana DEFLATE-algoritmia [RFC 1951, 1996] (eng. lossless data compression algorithm), joka on LZ77:n ja Huffmannin algoritmin [Huffman, 1952] yhdistelmä. Kyseistä DEFLATE-algoritmia on sovellettu myös ZIP-pakkausalgoritmissa sekä PNG-kuvatiedostoissa.

Alla on olevassa taulukossa on esitetty useita eri pakkausalgoritmeja sekä se, miten ne pakkaavat saman datan verrattuna toisiinsa. Taulukon 2 testissä käytettiin palvelinta Xeon X3450 VPS, jossa oli kaksi prosessoria ja 3 GB DDR3-muistia. Palvelin käytti CentOS 6.0 32bit OpenVZ -käyttöjärjestelmää. Testidata oli tar-pakattu tiedosto (1500 MB), joka sisälsi palvelimen ’/usr’-hakemiston (satunnaisia tiedostoja teksti- sekä binäärimuodossa). [Liu, 2011]

Metodi	Pakkausaika (sekuntia)	Purkuaika (sekuntia)	Pakkausnopeus (MB/s)	Purkunopeus (MB/s)	pakkaussuhde	pakkauskoko
zip	63.55	13.34	23.60	112.45	25.5%	392399
gzip	62.77	15.19	23.90	98.75	25.5%	392399
bzip2	482.80	74.08	3.11	20.25	22.7%	348500
pigz	31.34	8.47	47.86	177.10	25.5%	390946
pbzip2	157.51	24.15	9.52	62.11	22.7%	349177
lbzip2	90.50	32.64	16.57	45.95	22.7%	349180
lzip	558.20	37.98	2.69	39.50	17.3%	265793
plzip	164.15	30.19	9.14	49.69	17.9%	275013
7za 7z	454.72	31.97	3.30	46.92	15.6%	239439
7za bzip2	150.81	37.64	9.95	39.85	22.6%	347122

Taulukko 2: vertailu eri pakkaus ohjelmien suorituksessa (pakkaustaso 5) [Liu, 2011].

Taulukossa 3 on esitelty käytettyjen ohjelmien suhde toisiinsa. Testissä käytetyistä ohjelmista neljä ensimmäistä on yhdellä säikeellä toimivia ohjelmia ja kuusi muuta ovat neljän ensimmäisen jatkokehityksen tulosta, jotka käyttävät useampaa säiettä parantaakseen suorituskykyään. Taulukosta voidaan päätellä, että mikäli halutaan parasta mahdollista pakkaussuhdetta, ovat parhaat ohjelmat yhtä lukuun ottamatta monisäieversioita. Hyvälle pakkaussuhteelle tulee vääjäämättä korkea hinta pakkaukseen käytetyt ajan vuoksi. Samoin se tarvitsee paljon muistia pakkausta tehdessä.

Metodi	Versio	Lisätietoa
zip	v3.0	http://tools.ietf.org/html/rfc1951
gzip	v1.3.5	http://www.gzip.org/
bzip2	v1.03	http://www.bzip.org/
lzip	v1.1.3 rc1, LZMA algoritmi	http://lzip.nongnu.org/
pbzip2	v1.1.6, bzip2:n monisäie versio	http://www.bzip.org/
lbzip2	bzip2:n monisäie versio	http://lbzip2.org/
pigz	v2.1.6, gzip:n monisäie versio	http://zlib.net/pigz/
plzip	v0.80 rc1, lzip:n monisäie versio	http://lzip.nongnu.org/plzip.html
7za 7z	7zip(linux), 7z:n monisäe versio	http://www.7-zip.org/
7za bzip2	7zip(linux),bzip2:n monisäe versio	http://www.7-zip.org/

Taulukko 3: taulukossa 2 käytettyjen ohjelmien lisäselvennys.

Ohjelmistojakelijan kannalta tärkeää on saada kohtalainen pakkaustulos nopeasti. GZIP sekä ZIP ja PIGZ tarjoavat tämän ominaisuuden testin mukaan parhaiten. GZIP valittiin ohjelmistojakelijan pakkausalgoritmiksi, koska tämän tutkimuksen tekijällä on suurin käyttökokemus kyseisestä algoritmista. Koska kaikilla kolmella pakkausalgoritmilla (GZIP, ZIP, PIGZ) erot pakkauksessa sekä ajan käytössä olivat marginaaliset, on luontevaa valita algoritmi, jota on käytetty jo aiemmin. Tällöin uuden apukirjaston opettelemiseen ei mene aikaa, vaan ominaisuus voidaan lisätä sovellukseen nopeammin. Hakemistorakenteita kopioidessa asiakassovelluksille toimintaperiaate on, että haluttu hakemisto pakataan yhteen tiedostoon ennen lähetyksen aloittamista. Pakkauksen jälkeen yksittäinen tiedosto lähetetään asiakaspäätteille, joissa se puretaan ennalta määrättyyn paikkaan tiedonsiirron valmistuttua.

4.4. Datan lähetys ja vastaanottaminen

Datan lähetyksessä ja vastaanottamisessa tulee ottaa huomioon se, että asiakaspäätteet eivät välttämättä ole käyttöjärjestelmältään tai laitteiltaan samanlaisia. Joillakin verkkokortti voi esimerkiksi olla nopeampi kuin toisilla. Tämän takia on tärkeätä dataa lähettäessä ja vastaanottaessa, että palvelin pitää kirjaa jokaisen asiakassovelluksen kohdalla siitä, kuinka tiedonsiirto on toteutumassa. Tämän ominaisuuden tärkeyttä lisää myös ryhmälähetyksessä vaadittavat lisätarkistukset (datapakettien järjestäytyneisyys, sekä niiden hävinneisyys), jotta varmistutaan, että data, joka on siirretty asiakassovelluksiin, on täsmälleen sama kuin palvelimelta lähetetty data.

Palvelin käyttää datan tiedonsiirrossa ryhmälähetystekniikkaa, joka pohjautuu UDP-protokollaan sekä sen päälle rakennettuun sovellustason protokollaan. Sovellustason protokollan tehtävänä on pitää huolta siitä, että datapaketit tulevat asiakassovelluksille oikeassa järjestyksessä ja että mikäli

datapaketteja häviää matkalla, ne lähetetään uudelleen asiakassovelluksille. Lähetettävä data pakataan yhdeksi tiedostoksi, joka jaetaan ennen tiedonsiirron lähetystä moneksi pienemmäksi palaksi. Palat lähetetään asiakaspäätteille yksi kerrallaan niin kauan, että jokainen asiakaspääte on ilmoittanut saaneensa lähetetyn palan. Kun kaikki palat on lähetetty ja asiakaspäätteet ovat vastaanottaneet ne, palat kootaan takaisin tiedostoksi, joka puretaan jokaisella asiakaspäätteellä alkuperäiseen muotoonsa.

Datan tiedonsiirron lisäksi palvelin käyttää myös toista kommunikointikanavaa, jonka kautta palvelin lähettää käskyjä ja pyyntöjä asiakaspäätteille. Toisin kuin datan jakelukanavassa, tämä kanava käyttää TCP/IP-protokollaa, sillä asiakaspäätteiden tulee olla jatkuvassa yhteydessä palvelimeen ja asiakaspäätteen tulee olla tunnistettavissa palvelimelta. Käskyt ja pyynnöt ovat yksinkertaisia paketteja asiakkaan ja palvelimen välillä, jotka käyttävät TCP/IP-protokollalle ominaista yksi-yhdelle mekanismia.

4.5. Käytettävä ohjelmointikieli sekä apukirjastot

Ohjelmointikieliä, joita voidaan käyttää ohjelmistojakelijan toteutuksessa on useita. Esimerkiksi C, C++ sekä Java tukevat kaikki ryhmälähetystä ja ovat tarpeeksi voimakkaita ohjelmointikieliä sovelluksen toteutukseen. Kielen valintaan vaikuttaakin suuresti saatavilla olevat apukirjastot sekä riippuvuudet muihin sovelluksiin. Java on erittäin selkeä olio-ohjelmointia hyväksikäyttävä ohjelmointikieli, jota voidaan ajaa miltei millä tahansa alustalla tietokoneesta pesukoneeseen, toisaalta Javan vaatiman JRE:n (Java Runtime Enviroment) takia sen käyttö ohjelmistojakelijassa on epäkäytännöllisempää kuin käyttöjärjestelmässä natiivisti ajettavan sovelluksen käyttö.

Mikäli ohjelma toteutettaisiin Javalla, tulisi koneisiin olla asennettu JRE ennen kuin ohjelmistojakelijan asiakassovellusta voidaan käyttää. Sen takia koneissa olisi päivitettäviä ohjelmia yksi enemmän kuin muutoin. Mahdollisuus, että JRE lakkaa toimimasta tai jumittuu, on myös olemassa. JRE:n toiminnan lakkaaminen tarkoittaa samalla, että kaikki sovellukset, jotka käyttävät sitä alustanaan, lakkaavat toimivasta. Tästä huolimatta Javan vahvuus on nimenomaan mahdollisuus asentaa JRE hyvin monelle erilaiselle alustalle, jolloin myös ohjelmistojakelija toimisi käytännössä alustalla kuin alustalla ilman, että ohjelmistojakelijaa joutuisi uudelleen kääntämään kyseisille alustoille.

C++-ohjelmointikielen käyttäminen ohjelmistojakelijan kielenä ratkaisee riippuvuusongelman. C++-kääntäjä tuottaa käyttöjärjestelmässä ajettavia tiedostoja suoraan, eikä niiden suorittaminen vaadi kolmannen osapuolen sovellusta. Tämän takia ohjelmistojakelija on sidottu käyttöjärjestelmään, johon se on käännetty. Sovellus, joka on käännetty Windows-käyttöjärjestelmälle toimii vain kyseisessä käyttöjärjestelmässä, eikä esimerkiksi Linux tai Mac Os –käyttöjärjestelmissä. Vaikka tämä onkin työläämpää kuin Javaa käytettäessä, on se silti

kannattavampaa kokonaisuutta tarkastellessa. Tieturin koneluokissa on käytössä vain kaksi eri käyttöjärjestelmää, Windows sekä Linux, jolloin ohjelmistojakelija joudutaan kääntämään vain kyseisille käyttöjärjestelmille.

C++-kielelle on alunperin Trolltech ja sittemmin Nokia sekä Digia suunnitelleet voimakkaan apukirjaston nimeltä QT [Digia, 2013]. QT on erikoistunut käyttöliittymien toteutukseen, mutta tarjoaa myös kattavan C++-sovelluskehyksen sekä mahdollisuuden kääntää sama koodi (alustariippumaton ohjelmointi) niin Windows- kuin Linux-käyttöjärjestelmille ilman lisämuutoksia itse koodin. QT:n sovelluskehys tarjoaa ohjelmistojakelijan kannalta tärkeitä välineitä toteutukseen kuten esimerkiksi tiedostojen käsittelemiseen vaadittavat objektit, sql-tietokanta-yhteyden sekä tietokannan manipuloinnissa tarvittavat metodit. Näiden lisäksi QT tarjoaa kattavan TCP-palvelimen, jolla voidaan lähettää niin TCP/IP-paketteja kuin UDP-datagrammeja. Kun ohjelmistojakelija toteutetaan käyttäen apunaan QT-kirjastoa, voidaan minimoida uudelleen kirjoitettavan koodin määrä, Jos ohjelma käännetään eri käyttöjärjestelmille.

Palvelimen ja käyttöliittymän sekä palvelimen ja asiakassovelluksen välisessä käskynjako viestinnässä käytetään hyväksi WebSocket-protokollaa. QT-kirjastoa hyväksikäyttäen C++:lle on rakennettu WebSocketServer-apukirjasto [Ihalainen, 2011], jonka avulla palvelin voi keskustella kaikkien WebSocket-protokollaa tukevien ohjelmien, esimerkiksi web-selaimien kanssa. WebSocket protokollan avulla käyttöliittymä voidaan toteuttaa helposti ilman, että palvelimeen tarvitsee rakentaa tai liittää erikseen moduulia, joka ymmärtäisi palvelinpuolen web-teknologiaa. Käyttöliittymä kytkeytyy samaan käskynjakokanavaan kuin muutkin asiakaspäätteet.

4.6. Pääkomponentit

Ohjelmistojakelijan pääkomponenteiksi voidaan määritellä ne ohjelman osat, jotka voidaan eriyttää koko sovelluksesta fyysisesti eri paikkoihin tietoverkossa. Tällöin palvelinpuolen sovellus toimii itsenäisesti palvelimessa ja asiakaspuolen sovellus toimii itsenäisesti jokaisessa asiakaspäätteessä. Samoin käyttöliittymä palvelinsovelluksen hallintaan toimii web-selaimen kautta mistä tahansa sisäisenverkon sisällä.

4.6.1. Palvelinsovellus

Ohjelmistojakelijan palvelinsovellus on Windows Server -käyttöjärjestelmän päällä toimiva palvelu (eng. service), jonka tarkoituksena on olla käytettävissä kaikkina niinä aikoina, kun palvelin on toiminnassa. Palvelinsovellus pitää sisällään kolme eri tehtäväaluetta. Ensimmäisenä tehtäväalueena on, että palvelinsovellus pitää kirjaa kaikista asiakaspäätteistä, jotka ovat muodostaneet siihen yhteyden. Yhteys palvelimen ja asiakassovelluksen välillä on aina, kun asiakaspääte on esiasennettu sekä käyttöjärjestelmä on päällä. Palvelinsovelluksesta voidaan tarkastaa, mitkä asiakaspäätteet ovat toimintavalmiudessa ja mitkä eivät. Palvelinsovelluksen ei ole mahdollista herättää asiakaspäätteitä, mikäli ne ovat jostain syystä lopettaneet yhteydenpidon,

vaikka teknisesti siihen olisikin valmiudet Wake-On-Lan-muodossa. Palvelinsovellus sen sijaan peilaa senhetkisiä yhteyksiä aikaisempiin yhteyksiin ja pitää kirjaa, minkä nimiset asiakaspäätteet eivät ole yhteydessä, vaikka niistä on aikaisempi jälki tiedossa. Asiakaspäätteiden tiedot tallennetaan SQL-tietokantaan sekä niitä muutetaan reaaliajassa. Tämän avulla palvelinsovelluksella on viimeisin tieto asiakaspäätteistä, vaikka palvelin tai sovellus lopettaisi toimintansa ja jouduttaisiin käynnistämään uudelleen. Tietojen kerääminen ei varsinaisesti ole kriittinen osa-alue ohjelmistojakelijan toiminnassa, mutta jatkokehityksen kannalta se antaa uusia mahdollisuuksia.

Toisena palvelinsovelluksen tehtäväalueena on tiedostojen sekä hakemistorakenteiden pakkaaminen ryhmälähetystä varten. Pakkauksen tekeminen on hyvin vaativa operaatio palvelimelle. Se vaatii valtavasti tehoja palvelimen prosessorilta sekä kovalevyltä. Tästäkin huolimatta palvelinsovelluksen tulisi olla saavutettavissa myös pakkauksen aikana. Koska pakkaus suoritetaan palvelinsovelluksen sisällä ja se saattaa kestää useita minuutteja, on palvelinsovelluksen pystyttävä hyväksikäyttämään palvelimen käyttöjärjestelmän tukemaa moniajoa säikeiden avulla. Moniajo varmistaa, että jokaiselle ohjelmasäikeelle annetaan yksi kerrallaan vuoro toimia, jolloin tilannetta, jossa yksi ohjelma vie kaikki resurssit ja muut ohjelmat odottavat, ei tule tapahtumaan.

Kolmantena palvelinsovelluksen päätehtävänä on pakatun datan lähetys asiakassovelluksille. Datan pakkauksen jälkeen palvelinsovellus aloittaa ryhmälähetyksen niille asiakassovelluksille, jotka ovat ilmaisseet halukkuutensa kyseiseen dataan. Lähetyksen alussa palvelinsovellus lähettää MD5-tarkistussumman (eng. MD5 checksum) asiakassovelluksille sekä muita tarvittavia metatietoja lähetettävästä datasta. Tämän jälkeen pakattu data siirretään asiakassovelluksille. Metatietoa käytetään, jotta voidaan varmistua, että asiakassovelluksessa vastaanotettu data on yhdenmukainen palvelimelta lähetetyn datan kanssa.

Palvelinsovelluksen ja asiakassovelluksen sekä käyttöliittymän väliset pyynnöt, käskyt ja ilmoitukset toimivat WebSocket-protokollan avulla. Palvelinsovelluksessa ei ole erikseen eriytetty omaa kanavaa pelkälle käyttöliittymälle, vaan kaikki informaatio kulkee käskynjakokanavaa pitkin. Tämä mahdollistaa selkeämmän sekä joustavamman palvelinpuolen arkkitehtuurin. Mikäli käyttöliittymä käyttäisi omaa kanavaa, tulisi palvelinsovellukselle yksi TCP/IP-palvelinkomponentti lisää. TCP-palvelinkomponentit käyttävät itsessään jo omia säikeitä, mikä mahdollistaan samanaikaisen keskustelun ja datan lähettämisen asiakassovelluksien välillä. Tämä mahdollistaa myös käyttöliittymän käytön lähetyksen aikana sekä reaaliaikaisen seurannan, miten lähetys tai datan pakkaaminen edistyy.

Palvelinsovelluksen vikasietoisuutta kasvatetaan kahdella eri tavalla. Ensimmäinen varotoimi on, että palvelinsovellus on asetettu palveluksi palvelimelle. Tämä takaa palvelimen käyttöjärjestelmän

puolesta sen, että käyttöjärjestelmä käynnistää uudelleen palvelinsovelluksen, mikäli palvelinsovellus lopettaa toiminnan yllättäen. Mikäli palvelua ei saada uudelleen käynnistettyä tai siinä tapahtuu virheitä, käyttöjärjestelmä kirjaa ne ylös omaan lokikirjaan. Lokikirjan avulla pääkäyttäjä voi seurata mitä on tapahtunut ja korjata tilanteen. Toinen varotoimi on ns. ”Guard”-komponentti, joka pitää silmällä miten palvelinsovellus toimii. Mikäli palvelinsovellus jää loputtomaan silmukkaan tai TCP-palvelimet eivät vastaa Guardille tietyin väliajoin, Guard lopettaa palvelinsovelluksen toiminnan ja käynnistää sen uudelleen. Tällä varotoimella saadaan palvelinsovellus käynnistettyä uudelleen ja toimintavalmiuteen mikäli sovellus toimii vain osittain (itse palvelu toimii mutta kriittiset komponentit sen sisällä ei).

Palvelinsovelluksesta on mahdollista lähettää yksinkertaisia komentoja jokaiselle asiakaspäätteelle kuten esimerkiksi ”käynnistä setup.exe”, ”käynnistä kone uudelleen” tai ”sammuta kone 120 sekunnin kuluttua”. Tämä nopeuttaa esimerkiksi ohjelman asennuksen aloittamista ja viimeistelyä sen jälkeen kun asennusmedia on ladattu onnistuneesti jokaiselle asiakaspäätteelle. Vaihtoehtoisesti pitäisi asentajan asennusmedian kopioinnin jälkeen esimerkiksi käydä jokainen asiakaspääte yksitellen lävitse ja käynnistää ”setup.exe”-toiminto manuaalisesti.

4.6.2. Asiakassovellus

Asiakassovellus on Windows- sekä Linux-käyttöjärjestelmissä toimiva sovellus, jonka tehtävänä on vastaanottaa palvelinsovellukselta lähetetty data. Asiakassovellus käynnistyy automaattisesti, kun asiakaspäätteen käyttöjärjestelmä käynnistyy. Tällä tavoin voidaan pitää huolta, että jokainen käynnissä oleva asiakaspääte on yhteydessä palvelinsovellukseen ja valmis ottamaan vastaan käskyjä palvelimelta. Toisin kuin palvelinsovellus asiakassovellusta ei aseteta palveluksi. Syynä tähän on Windowsin ja Linuxin erikaltaiset palvelujen toteutustavat. Myös tapa, jolla Windows kloonataan Tieturi Oy:ssä, hankaloittaa palveluksi rekisteröitymistä. Jotta ohjelma voidaan rekisteröidä palveluksi, täytyy sille antaa järjestelmävalvojatason oikeudet, jolla palvelu käynnistetään. Kursseista riippuen asiakaspäätteet voivat olla liitettyinä Windows-toimialueeseen tai ne voivat olla työryhmässä. Mikäli asiakaspäätteet ovat toimialueessa, niille pakotetusti syötetään uusia järjestelmänvalvoja tunnuksia sekä mahdollisesti poistetaan käytöstä vanhoja asennuksessa tehtyjä tunnuksia. Tästä syystä palvelulle on hankala määritellä yhtä järjestelmänvalvoja tunnusta, jolla se voisi käynnistyä asiakaspäätteen verkkokonfiguraatiosta huolimatta. On järkevämpää käynnistää sovellus automaattisesti, kun käyttäjä kirjautuu sisään, jolloin ohjelma avautuu sen hetkisen käyttäjän oikeuksilla. Asiakassovellus ei tarvitse toimintaansa järjestelmätason tunnuksia, joten ei ole suurtakaan väliä millä oikeuksilla sen hetkinen käyttäjä on kirjautuneena sisään.

Asiakassovellus käynnistyttyään piiloutuu käyttäjältä ikoniksi ilmaisinalueelle (eng. system tray). Asiakassovellus ei kopioinnin tai datan purkamisen aikana ilmoita käyttäjälle erikseen mitään, vaan

asiakassovelluksen käyttöliittymä on avattava ilmaisinalueen ikonista. Tämä johtuu siitä, että mikäli halutaan kopioida tiedostoja tietokoneiden välillä silloin, kun esimerkiksi koulutus on käynnissä ja koneella on oppilas, on prosessin oltava huomaamaton. Mikäli asiakassovellus ilmoittaisi esimerkiksi virhetilanteista tai onnistuneesta tiedoston siirrosta, on mahdollista että kurssin seuraaminen häiriintyisi.

Joissakin asennustilanteissa on kuitenkin suotavaa, että voidaan vaihtaa asiakassovelluksen asetuksia. Esimerkiksi hakemisto, johon data halutaan purkaa, pitää olla vaihdettavissa. Myös osoite, jossa palvelinsovellus toimii, pitää olla vaihdettavissa kuten myös datanjakelukanavan nimi. Muut siirtämiseen sekä purkamiseen liittyvät asetukset tulevat palvelinsovelluksesta käskynjakokanavan kautta metadatana ennen siirron alkua tai sen jälkeen.

Kuten palvelinsovelluksessakin on asiakassovelluksessa myös kaksi TCP-palvelinta toiminnassa: toinen käskynjakokanavaa varten ja toinen datanjakokanavaa varten. Erona palvelin- ja asiakassovelluksen välillä on, että palvelinsovelluksen ei tarvitse reagoida siirron yhteydessä mihinkään muuhun kuin itse siirtoon. Datan siirron aikana käskynjakokanavan on toiminnassa, mutta asiakassovellus ei reagoi sieltä tulleisiin viestiin ennen kuin datan siirtoprosessi on valmis.

Teoriassa datan siirtonopeus voi vaihdella suuresti riippuen asiakaspäätteiden laitekokoonpanosta, jolloin asiakaspäätteellä tulisi olla mahdollisuus ilmoittaa palvelinsovellukselle, että palvelimen lähettämän datan määrä on liian suuri. Esimerkiksi Symantec Ghost määrittelee lähetyksen alussa niin sanotun Master Clientin (MC), joka on hitain ryhmälähetykseen kuuluva asiakaspääte. Tällöin datan siirtonopeus, jolla Symantec Ghost Server lähettää dataa, on kiinnittynyt MC:n nopeuteen [Symantec, 2007]. Ohjelmistojakelija ei tällaista varotoimea sisällä vaan palvelinsovellus lähettää tietoa sille määrätyllä maksiminopeudella. Syynä tähän on Tieturi Oy:ssä standardoitu konekanta, jonka johdosta luokkien koneet ovat samaa mallia sekä samalla laitekokoonpanolla varustettuina. Tällöin on oletettavaa, että luokan sisällä jokaisen asiakaspäätteen nopeus on samankaltainen. Mikäli asiakaspääte jää jälkeen palvelimen datansiirrosta, asiakas ilmoittaa siitä suoraan palvelimelle, jolloin palvelinsovellus jää odottamaan kyseistä asiakaspäätettä kunnes se on valmis jatkamaan datan siirtoa muiden asiakaspäätteiden tahdissa.

5. Ohjelmistojakelijan arkkitehtuuri

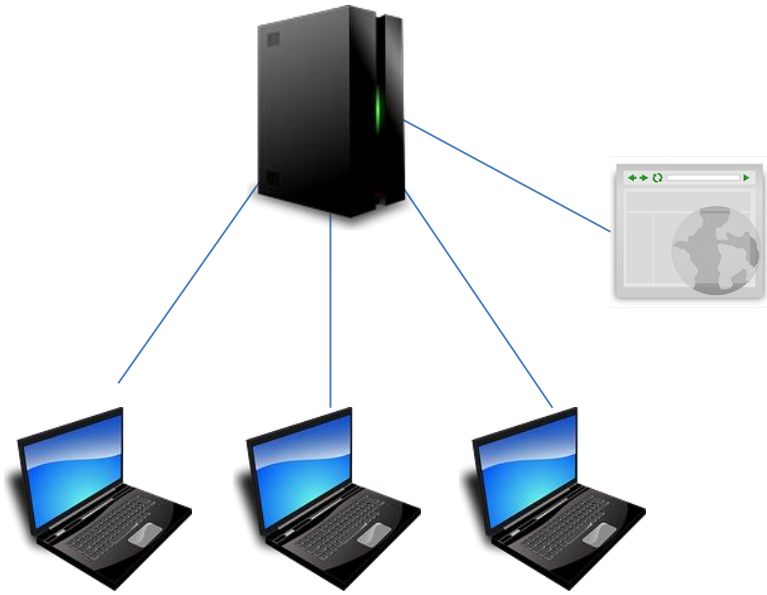
Tässä luvussa esitetään miten ohjelmistojakelija on toteutettu edellisen luvun vaatimusmäärittelyiden ja ohjeiden perusteella. Ohjelmistojakelijan arkkitehtuurin yleiskuvas sekä asiakassovelluksen että palvelinsovelluksen ja käyttöliittymän arkkitehtuuri esitellään kohdassa 5.1. Asiakassovelluksen ja palvelinsovelluksen välisessä kommunikaatiossa käytetyt protokollat avataan tarkemmin kohdassa 5.2.

Arkkitehtuurikaaviot on toteutettu käyttämällä UML2-notaatiota [OMG.org, 2005]. Sinisellä pohjalla olevat laatikot kuvaavat kokonaisuuksia (eng. package), joiden komponentit kuuluvat tiettyyn kokonaisuuteen. Laatikot, joiden yläkulmassa on kuvake, esittävät komponentteja. Komponentit ovat niitä osia järjestelmän arkkitehtuurista, jotka ovat ratkaisevia ohjelman toimivuuden kannalta. Komponentti voi olla koodissa objektin ilmentymä tai laajempi kokonaisuus, joka pitää sisällään uusia komponentteja. Kolmiulotteiset laatikot kuvaavat ulkoista palvelua tai abstraktia käsitettä, kuten esimerkiksi käyttöjärjestelmän tiedostojärjestelmää, toista palvelinta tai kommunikaatiokanavaa. Komponenttien välisiä yhteyksiä kuvataan joko katkoviivalla tai viivoilla, joita yhdistää pallo. Katkoviivalla yhdistyneet komponentit kuvaavat tilaa, jossa toinen komponentti käyttää hyväkseen suoraan toista komponenttia ilman rajapintaa. Pallolla yhdistetyt komponentit kuvaavat taasen rajapintojen kautta tapahtuvaa yhteyttä.

5.1. Arkkitehtuuri

Ohjelmistojakelijan yleisarkkitehtuuri perustuu asiakas-palvelin –arkkitehtuurimalliin (eng. client-server architecture). Tässä mallissa asiakas ottaa yhteyden palvelimeen, joka jakaa resurssejaan asiakkaalle; ohjelmistojakelijan tapauksessa resurssit ovat esimerkiksi pakattuja datatiedostoja. Koska palvelinsovellus hoitaa niin palvelinpuolen sovelluslogiikan kuin myös tietokantayhteydet kutsutaan tätä asiakas-palvelin arkkitehtuuria tarkennetusti ”tason kaksi” malliksi. [Gallaugher & Ramanathan, 1996]. Asiakkaan yhteys palvelimeen on jaettu kahteen eri kanavaan, joita käyttäen asiakkaalle siirretään dataa sekä metatietoja.

Käyttöliittymän suhdetta ohjelmistojakelijan palvelinsovellukseen voidaan pitää MVC-arkkitehtuurimallin kaltaisena [GOF, 1994]. Käyttöliittymä on näkymä, joka saa tietonsa mallilta, tässä tapauksessa palvelinsovelluksen tietokannasta, ja jonka kontrollerina toimii palvelinsovelluksen sovelluslogiikka. [GOF, 1994]



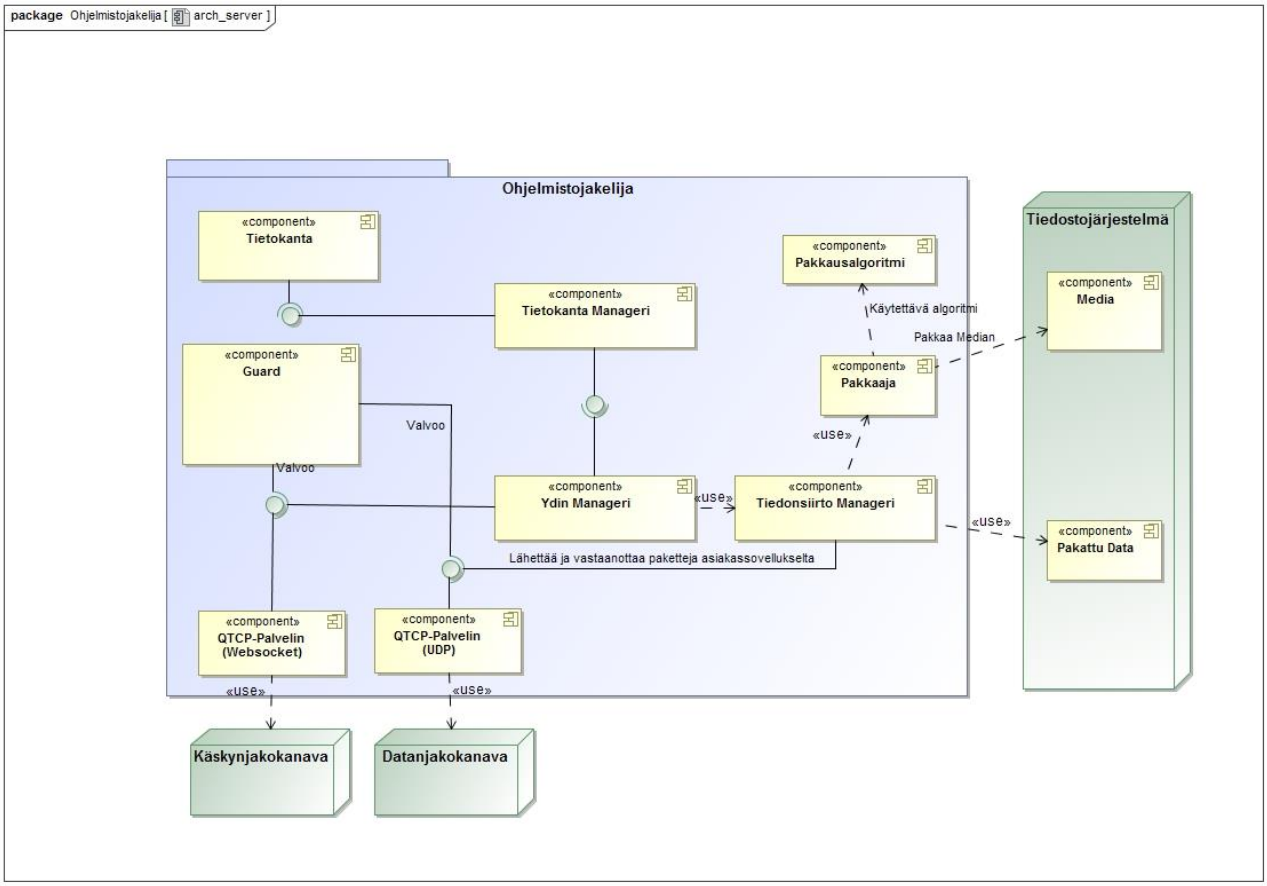
Kuva 7: Ohjelmistojakelijan yleisarkkitehtuuri.

Kuvassa 7 on esitetty pelkistetty yleisarkkitehtuurikaavio, jonka tarkoituksena on havainnollistaa eri pääkomponenttien suhde toisiinsa. Käyttöliittymä ja asiakassovellukset keskustelevat palvelimen kanssa, mutta eivät suoraan toistensa kanssa. Kun kaikki informaatio kulkee palvelinsovelluksen kautta, pystyy palvelinsovellus pitämään kirjaa tehokkaasti tapahtumista ja säätelemään viestien kulkua koko sovelluksen sisällä sekä lähettämään käyttöliittymälle ajantasaista tietoa järjestelmän tilasta.

5.1.1. Palvelinsovellus

Palvelinsovellusta suunniteltaessa huomio on kiinnitetty arkkitehtuurin yksinkertaistamiseen, toimivuuteen ja turvallisuuteen. Edellä mainitut laatukriteerit ovat tiukasti kytköksissä toisiinsa, sillä mikäli arkkitehtuuri on monimutkainen, on mahdollista, että sovellusta toteuttaessa ilmenee ongelmia arkkitehtuuripäätösten vuoksi. Monimutkainen arkkitehtuuri tarkoittaa useasti myös, että se on raskaampi ylimääräisten komponenttikutsujen vuoksi kuin mitä sama sovellus olisi, jos se olisi suunniteltu yksinkertaisemmin. Sovelluksen turvallisuuskriteeri tyydytetään ensinnäkin käyttämällä komponenttia, joka valvoo muiden komponenttien tilaa. Tämän lisäksi turvallisuuskriteeri otetaan huomioon käyttöjärjestelmä tasolla, jossa käyttöjärjestelmä valvoo koko sovelluksen tilaa

Kuvassa 8 on esitetty palvelinsovelluksen kokonaisarkkitehtuuri, jonka perusteella ohjelmistojakelijan palvelinsovellus on toteutettu. Kuvassa on esitetty pääkomponentit sekä niiden suhde toisiin komponentteihin ja järjestelmiin (tiedostojärjestelmä, käskynjakokanava, datanjakokanava).



Kuva 8: Palvelinsovelluksen arkkitehtuuri.

Ohjelmistojakelijan tärkeimmät komponentit ovat nimetty managereiksi. Niiden tehtävä on valvoa ja suorittaa kriittisiä toimintoja sovelluksen sisällä. Tällaisia ovat esimerkiksi asiakassovelluksien käskyttäminen, tiedonsiirto sekä tietokannan käyttäminen. Ohjelmajakelijan päätoiminnallisuus - sovelluslogiikka - on keskitetty Ytimeen (Kuva 8). Sen tehtävä on pitää kirjaa siitä, mitä sovellus itse tekee ja missä tilassa asiakassovellukset ovat. Käyttöliittymäkäskyt tulevat suoraan Ytimeen, joka tulkitsee käskyt ja toimii niiden mukaisesti. Ydin on yhteydessä kahteen muuhun manageriin, Tietokantaan ja Tiedonsiirtoon, sekä käskynjakokanavaan. Näitä komponentteja hyväksikäyttäen Ydin voi hallinnoida koko sovelluksen toimintaa sekä toteuttaa sille annettuja käskyjä.

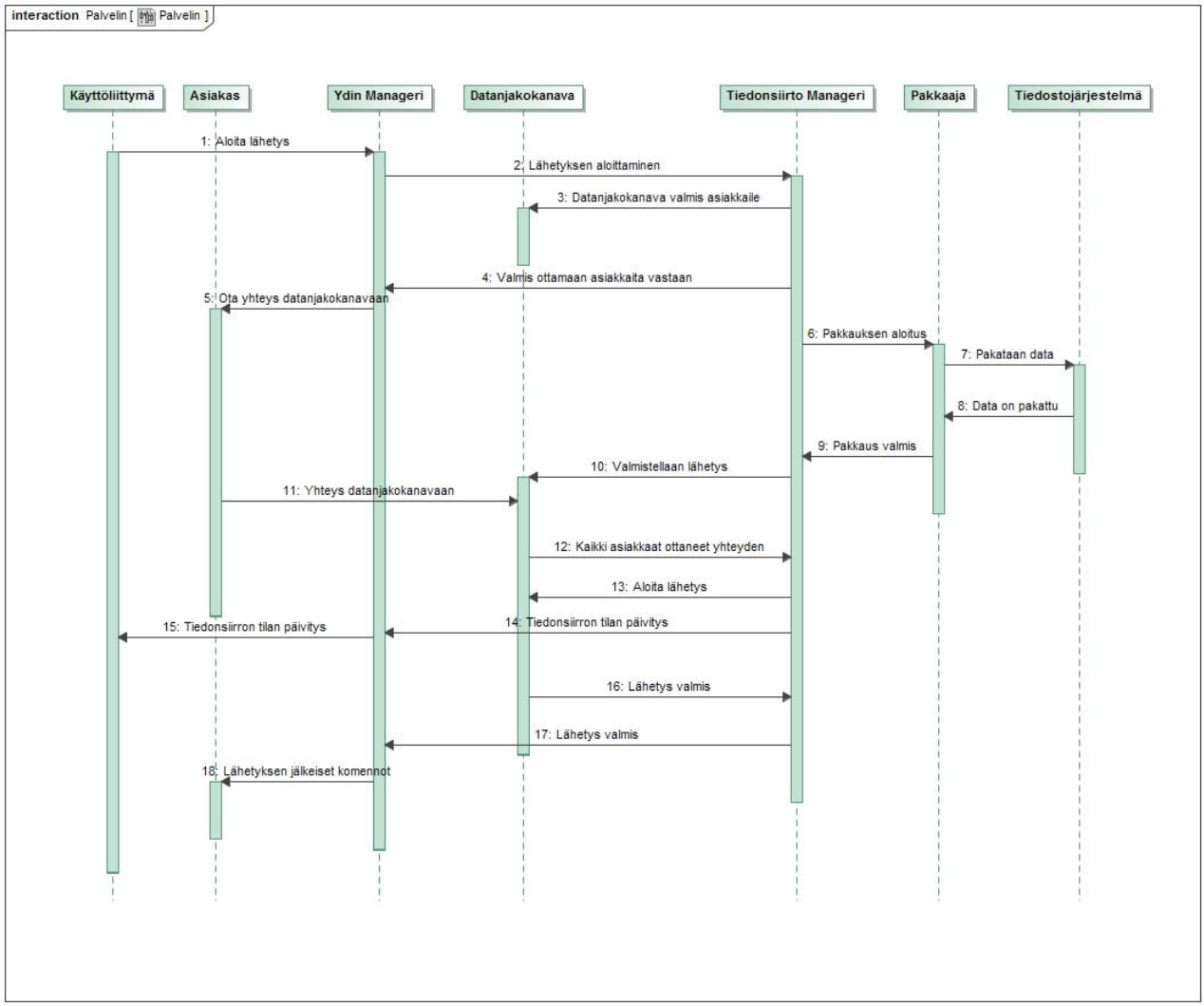
Tiedonsiirron tehtävä on valmistella sekä aloittaa tiedonsiirto palvelinsovelluksen ja asiakassovelluksien välillä sekä valmistella haluttu data lähetystä varten. Kun Tiedonsiirto saa Ytimeltä käskyn aloittaa tiedonsiirto tiettyjen asiakkaiden kanssa, aloittaa se halutun datan pakkaamisen käyttäen hyväkseen Pakkaaja-komponenttia. Pakkaaja-komponentti hyödyntää kuvan 8 mukaisesti pakkausalgoritmi-komponenttia, jota käyttäen se pakkaa halutun hakemistorakenteen tai tiedoston ja siirtää sen väliaikaisvarastoon tiedonsiirron ajaksi. Tiedonsiirto saa Pakkaajalta tiedon, kun pakkaus on valmistunut. Tämän jälkeen Tiedonsiirto valmistelee lähetyksen ja odottaa

asiakassovelluksien yhteydenottoa. Kun kaikki asiakkaat ovat ottaneet yhteyden, Tiedonsiirto aloittaa lähetyksen näiden asiakassovelluksien kanssa. Lähetyksen valmistuttua Tiedonsiirto ilmoittaa Ytimelle lähetyksen tilan eli onko lähetys onnistunut vai epäonnistunut.

Tietokanta-managerin tehtävänä on toimia rajapintana tietokannan ja Ytimen välillä. Ydin antaa kirjoitus- sekä lukukäskyjä Tietokannalle, joka muuttaa käskyt tietokantaan sopiviksi hakukomennoiksi. Ydin voisi olla suoraan yhteydessä tietokantaan, mutta on tulevaisuutta ajatellen mielekästä käyttää ns. adapteria tietokannan ja tietokantaa käyttävän komponentin välillä. Mikäli ohjelmistojakelijaa halutaan kehittää eteenpäin tai mikäli tietokantasovellus muuttuu, voidaan uusi tietokanta ottaa käyttöön päivittämällä adapteria. Tällöin jo ennestään paljon koodia sisältävää Ydintä ei tarvitse muuttaa, vaan riittää, että adapteri vaihdetaan tai päivitetään toimimaan uuden tietokannan kanssa.

Guard on turvallisuuden ja toimivuuden kannalta tärkeä komponentti ohjelmistojakelijassa. Sen tehtävänä on pitää kirjaa käskynjako- sekä datanjakopalvelimien toiminnasta. Guard käyttää QTCP-palvelimien tarkkailussa syke-mekanismia (eng. heartbeat). Syke-mekanismi toimii siten, että komponentti lähettää toiselle komponentille tietyin väliajoin Ping-paketin, johon vastaanottava komponentti vastaa toisella Pong-paketilla. Jos vastaanottava komponentti ei vastaa Ping-pakettiin tietyn ajan sisällä, voidaan olettaa, että kyseinen komponentti ei ole toiminnassa tai komponenttiin kohdistuva kuorma on niin suuri, että sen toiminta on hidastunut merkittävästi. Mikäli Guard huomaa, että jompikumpi palvelimista ei vastaan sen tasaisin väliajoin lähetettyihin ACK-paketteihin, raportoi Guard-komponentti palvelimen virhetilasta Ytimelle sekä käynnistää virhetilaan joutuneen palvelimen tai viimetilassa koko palvelinsovelluksen uudelleen.

QTCP-palvelimet (Kuva 8) ovat fyysisiä QT-kirjaston objekteja, joiden avulla voidaan luoda niin UDP- kuin TCP-palvelimia. Datanjakokanava käyttää UDP-protokollaa palvelimessa ja käskynjakokanava käyttää TCP/IP-protokollaa palvelimessa. Näiden palvelimien tarkoitus on mahdollistaa samanaikainen kommunikaatio ja tiedonsiirto palvelinsovelluksen ja asiakassovelluksien välillä. Jokainen asiakassovellus ja käyttöliittymä ottavat yhteyden palvelinsovelluksen Käskynjakokanavaan WebSocket-protokollan avulla. Tätä kanavaa pitkin Ydin ilmoittaa käyttöliittymässä määrätyille asiakkaille, milloin asiakassovellukset voivat ottaa yhteyden datanjakokanavaan tiedonsiirron aloitusta varten. Yhteys muodostetaan tiedonsiirtokanavassa käyttämällä istuntoja. Kun asiakassovellus ottaa yhteyden datanjakelukanavaan, lähettää se istunnon tunnisteen palvelinsovellukselle, jolloin Tiedonsiirto pystyy määrittelemään, mitä dataa asiakassovellus haluaa vastaanottaa, mikäli samanaikaisia tiedonsiirtoja on useita. Tällaiset tilanteet ovat poikkeuksia mutta mahdollisia. Istuntotunnisteita käyttämällä voidaan myös suojata tiedonsiirtoa ulkopuolisilta hyökkäysyrityksiltä. Mikäli jokin asiakassovellus ottaa yhteyden palvelinsovellukseen ilman mitään istuntotunnistetta, hylätään yhteydenotto kokonaan.



Kuva 9: Tiedonsiirron aloituksen sekvenssikaavio.

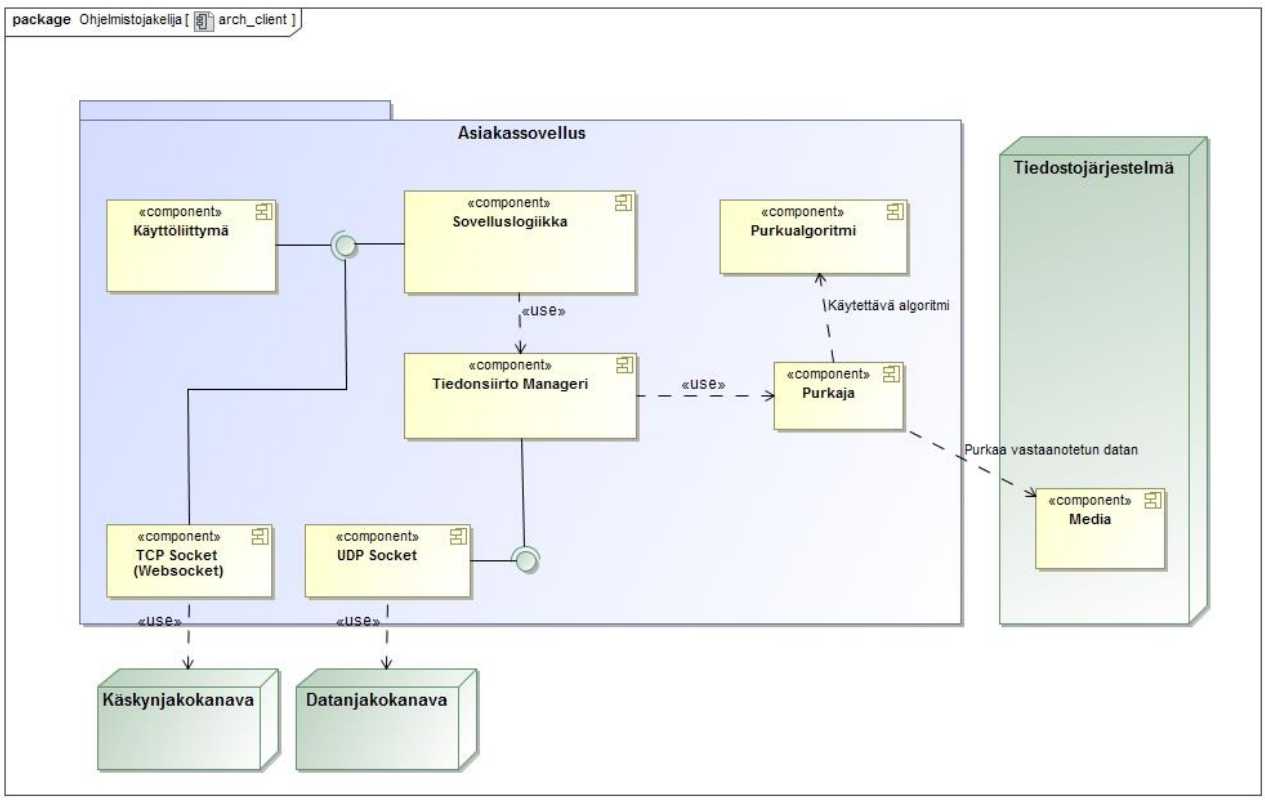
Kuvan 9 sekvenssikaaviossa havainnollistetaan vaiheet, joiden jälkeen tiedonsiirto voidaan aloittaa palvelinsovelluksen ja asiakassovelluksien välillä, kun palvelinsovellus saa käskyn käyttöliittymältä suorittaa tiedonsiirto tiettyjen asiakassovellusten kanssa. Sekvenssikaaviossa käyttöliittymä antaa käskyn käskynjakokanavaa pitkin Ytimelle aloittaa tiedonsiirto tiettyjen asiakassovellusten kanssa. Ydin kirjaa tapahtuman ylös ja siirtää hallinnan Tiedonsiirrolle. Tiedonsiirto valmistelee istunnon datanjakokanavassa ja ilmoittaa Ytimelle milloin istunto on valmis vastaanottamaan asiakassovelluksia. Samaan aikaan, kun asiakassovellukset ottavat yhteyden kyseiseen istuntoon, valmistelee Tiedonsiirto halutun datan (Kuva 9, kohta 6: pakkaus, siirto väliaikaiseen varastoon) lähetystä varten. Datan valmistuttua lähetystä varten, sekä kaikkien asiakassovellusten otettua yhteys tai kun määritelty aikakatkaisu umpeutuu, aloitetaan tiedonsiirto (Kuva 9, kohta 13). Tiedonsiirto lähettää Ytimelle tilatietoja tiedonsiirrosta, jotta Ydin pystyy välittämään kyseiset

tiedot käyttöliittymälle. Tiedonsiirto ilmoittaa Ytimelle oliko tiedonsiirto onnistunut vai epäonnistunut (Kuva 9, kohta 17). Ydin lähettää onnistuneen tiedonsiirron jälkeen käskynjakokanavaa pitkin komennot, joita halutaan ajaa jokaisella asiakassovelluksella (Kuva 9, kohta 18).

5.1.2. Asiakassovellus

Asiakassovelluksen arkkitehtuuri on hyvin samankaltainen mutta pelkistetympi versio palvelinsovelluksen arkkitehtuurista. Suurimpina eroina ovat Guard-komponentin ja tietokannan puuttuminen sekä käyttöliittymän integrointi suoraan sovellukseen itseensä. Asiakassovelluksen, kuten myös palvelinsovelluksen, arkkitehtuuri on suunniteltu mahdollisimman läpinäkyväksi käyttäjälle. Tämä tarkoittaa sitä, että päätelaitteen loppukäyttäjän ei tulisi huomata sovelluksen toimintaa laisinkaan.

Asiakassovellus käyttää kuutta eri prosessia toiminnassaan. Niiden tarkoituksena on jakaa asiakassovelluksen tuottamaa hetkellistä kuormaa päätelaitteen prosessoreiden kesken (mikäli päätelaitteessa on useampi kuin yksi prosessori). Sovelluslogiikalla, Pakkaajalla, Tiedonsiirrolla, käyttöliittymällä sekä kummallakin Socket-komponentilla on käytössään oma prosessi. Esimerkiksi dataa purettaessa muu asiakassovelluksen toiminnallisuus ei jää odottamaan purkamisen valmistumista, vaan pystyy toimimaan samanaikaisesti.

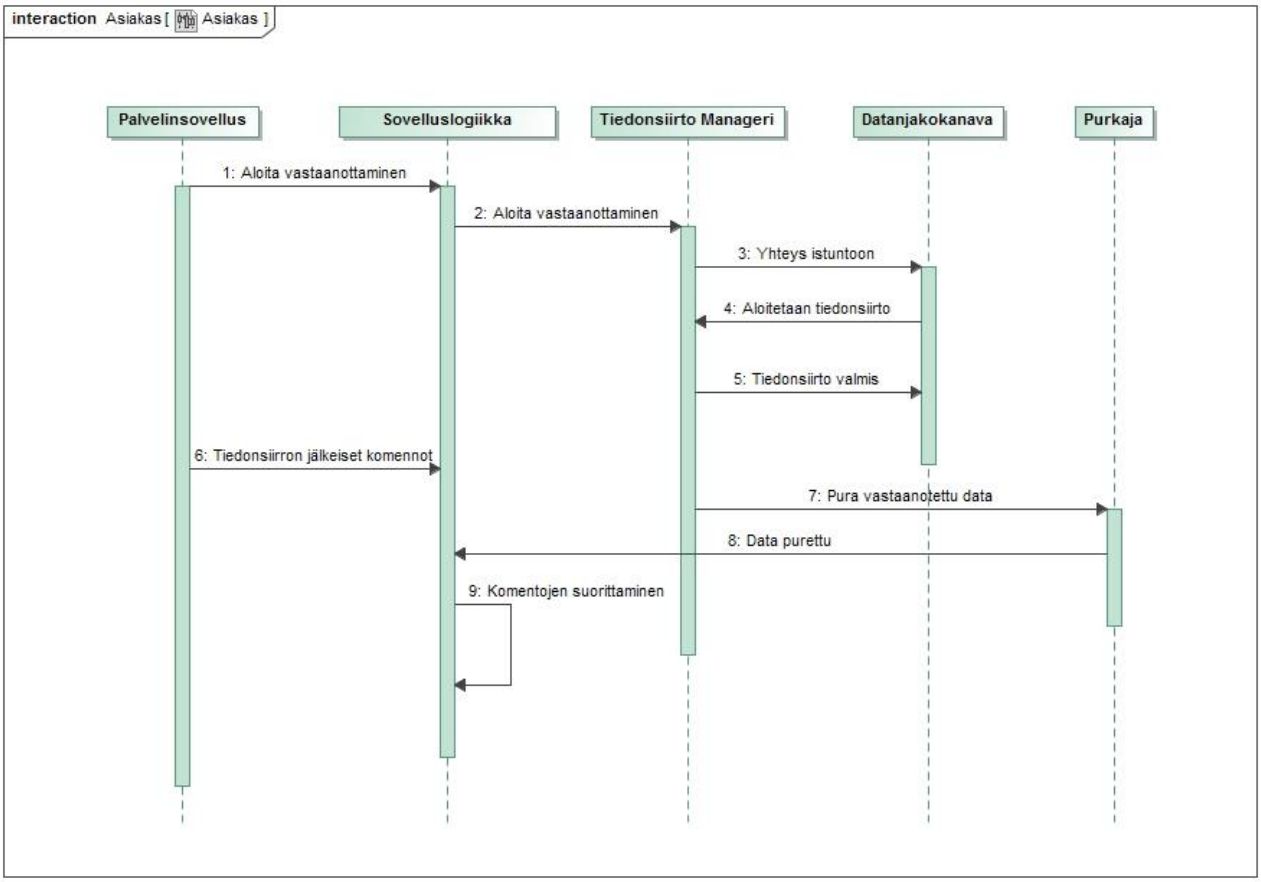


Kuva 10: Asiakassovelluksen yleisarkkitehtuuri.

Kuvassa 10 on esitetty asiakassovelluksen yleisarkkitehtuurin pääkomponentit sekä niiden kytkökset toisiinsa. Sovelluslogiikka vastaa palvelinsovelluksen Ydintä (Kuva 8). Sen tarkoituksena on toimia niin käyttöliittymän mallina kuin muun toiminnallisuuden sovelluslogiikkana. Käyttöliittymä on suunniteltu MVC-mallin mukaisesti, jossa käyttöliittymä-komponentti on näkymä, jonka kontrollerina sekä mallina toimii Sovelluslogiikka-komponentti. Sovelluslogiikka on suorassa yhteydessä käskynjakokanavaan, jonka kautta palvelinsovellus lähettää käskyjä asiakassovellukselle, sekä jonka kautta asiakassovellus raportoi omasta tilastaan palvelinsovellukselle.

Tiedonsiirto (Kuva 10) toimii samankaltaisesti kuin palvelinsovelluksen arkkitehtuurissakin. Sen tehtävänä on ottaa yhteys Sovelluslogiikalta saamaansa istuntoon ja vastaanottaa palvelimelta lähetetty data. Kun Tiedonsiirto on saanut Sovelluslogiikalta käskyn aloittaa tiedonsiirto, ottaa se yhteyden datanjakokanavan istuntoon ja odottaa kunnes palvelinsovellus on valmis lähettämään datan. Siirron valmistuttua Tiedonsiirto antaa käskyn Purkaja-komponentille purkaa vastaanotettu data ennalta määrättyyn hakemistoon tiedostojärjestelmässä.

Kuvassa 8 on esitetty Purkajan käyttävän hyväkseen purkamisessa purkualgoritmi-komponenttia. Syynä purkualgoritmin erotteluun Purkaja–komponentista on se, että purkualgoritmeja voi olla useita. On perusteltua mahdollistaa Purkaja-komponentin käyttää useita eri purkualgoritmeja. Mikäli tahdotaan vaihtaa purkualgoritmia, ei Purkajaan tarvitse tehdä muutoksia, vaan Purkajaa voidaan käskeä käyttää eri purkualgoritmi-komponenttia. Sama pätee myös palvelinsovelluksen Pakkaajaan ja pakkausalgoritmiin. Esimerkiksi jos data on jo valmiiksi pakatussa muodossa, ei ole järkevää pakata pakattua dataa uudelleen. Tällöin Purkajaa ja Pakkaajaa voidaan käskeä olla käyttämättä mitään algoritmia, jolloin pakkaus- ja purku-vaiheet tiedonsiirrossa jäävät pois ja tiedonsiirtoon käytetty kokonaisaika lyhenee huomattavasti.



Kuva 11: Tiedonsiirron sekvenssikaavio asiakassovelluksessa.

Kuvassa 11 on esitetty tiedonsiirron vaiheet sekvenssikaaviona asiakassovelluksen näkökulmasta. Kun asiakassovellus saa palvelinsovelluksen käskyn käskynjakokanavan kautta aloittaa tiedonsiirto (Kuva 11, kohta 1), välittää sovelluslogiikka yhteyspyynnön Tiedonsiirrolle, joka ottaa yhteyden istuntoon datanjakokanavassa (Kuva 11, kohta 3). Palvelin aloittaa tiedonsiirron, kun kaikki tiedonsiirtoon määritetyt asiakkaat ovat ottaneet yhteyden istuntoon. Tiedonsiirron valmistuttua (Kuva 11, kohta 5) Tiedonsiirto antaa Purkajalle käskyn purkaa vastaanotettu data (Kuva 11, kohta 7) ja siirtää se ennalta määrättyyn hakemistoon. Kun data on purettu, sovelluksen hallinta palaa Sovelluslogiikalle (Kuva 11, kohta 8), joka ajaa palvelinsovellukselta saadut tiedonsiirron jälkeiset komennot siinä järjestyksessä kun ne on annettu (Kuva 11, kohta 9).

Sovelluslogiikan, Tiedonsiirron sekä Purkajan elinkaaret sekvenssikaaviossa ovat yhtä pitkät. Tämä tarkoittaa sitä, että niiden toiminta ei lopu eikä odota toisia komponentteja, vaan kaikki toimivat samanaikaisesti riippumatta toisistaan. Näin ollen, mikäli esimerkiksi palvelinsovellukselta tulee käsky lopettaa tiedonsiirto tiedonsiirron ollessa käynnissä, pystyy asiakasohjelma toteuttamaan annetun käskyn heti ilman, että se joutuisi odottamaan toisen komponentin toiminnan loppumista.

5.2. Sovelluksien välinen kommunikaatio

Sovelluksilla on käytössään kaksi eri kommunikointikanavaa. Ensimmäinen on käskynjakokanava, jonka tehtävä on välittää palvelimen, käyttöliittymän ja asiakaspäätteiden välisiä viestejä. Toinen on datanjakokanava, jonka tehtävä on toimia välittäjänä asiakassovellusten ja palvelimen tiedonsiirrossa. Kanavien ratkaisevat erot ovat protokollissa, joita ne käyttävät tiedon lähettämisessä ja vastaanottamisessa. Seuraavassa on esitelty kanavissa käytettyjen protokollien toiminta sekä rakenne.

Ohjelmistojakelijan palvelinsovellus käyttää WebSocket-tekniikkaa käyttöliittymän, palvelimen ja asiakassovellusten välisessä kommunikaatiossa. Näin ollen palvelinsovellukseen ei erikseen tarvitse rakentaa komponenttia, jonka kautta web-sivut ovat yhteydessä palvelimeen vaan käyttöliittymä voi keskustella palvelimen kanssa samalla käskynjakokanavalla kuin muutkin asiakassovellukset. Käyttöliittymä lähettää tietopyyntöjä sekä käskyjä WebSocketin avulla suoraan ohjelmistojakelijan Ydin-komponentille, joka käsittelee halutut pyynnöt ja käskyt ja vastaan niihin.

Datanjakokanava käyttää UDP-protokollaa sen nopeuden sekä ryhmälähetyksen käyttömahdollisuuden vuoksi. UDP-protokollan päälle rakennetaan sovellustason protokolla, jonka tehtävänä on varmistaa, että jokainen asiakas saa lähetetyt paketit sekä sen, että paketit tulevat oikeassa järjestyksessä asiakassovelluksille. Protokollia, jotka toteuttavat pakettien saapumis- ja oikeellisuustarkistukset on olemassa useita, esimerkiksi Pragmatic General Multicast (PGM), mutta ohjelmistojakelijan tapauksessa on mielekästä suunnitella protokolla alusta alkaen ohjelmistojakelijaan sopivaksi.

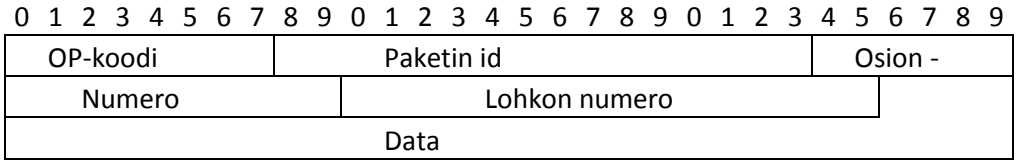
Ohjelmistojakelijalle kehitettyä protokollaa kutsutaan Reliable Ordered Multicast -protokollaksi (ROM). ROM-protokollan toiminta jaetaan neljään eri vaiheeseen: ensimmäinen vaihe on ryhmälähetys istunnon alustaminen ja asiakkaiden vastaanottaminen, toinen vaihe on tiedonsiirto palvelimen ja asiakassovelluksien välillä, kolmas lähetetyn datan oikeellisuuden varmistus ja neljäs istunnon lopettaminen.

Ensimmäisessä vaiheessa istunto alustetaan Tiedonsiirron avulla (Kuva 8). Asiakkaat ottavat yhteyden istuntoon käskynjakokanavalla saatujen tietojen perusteella. Kun kaikki asiakkaat ovat yhdistyneet, varmistetaan asiakkaiden valmius alkaa vastaanottaa dataa lähettämällä asiakassovelluksille paketti, johon ne vastaavat mikäli ovat valmiina. Jos vastausta ei saada asiakkailta määritellyn ajan sisällä, tiputetaan kyseiset asiakkaat pois tiedonsiirrosta ja siirrytään protokollan toiseen vaiheeseen.

Toisessa vaiheessa palvelimelle pakattu data lähetetään asiakassovelluksille ryhmälähetyksenä. Datan lähetysvaiheessa ensimmäisenä lähetetään jokaiselle asiakassovellukselle metatietoa

lähetyksestä. Metatieto sisältää lähetettävän tiedoston nimen, MD5-tarkistussumman sekä lähetettävien osioiden ja osioissa olevien lohkojen määrän. Lähetettävä data jaetaan osioihin, jotka jaetaan lohkoihin. Osioiden lukumäärä määräytyy lähetettävän datan koosta, mutta lohkojen määrä osioissa on vakio. Jakamalla data kyseisellä tavalla mahdollistetaan datan uudelleen lähettäminen asiakkaille myös niissä tilanteissa, joissa datapaketti on hukkunut matkalla yhden osion tarkkuudella. Datan lähetys jatkuu itse tiedonsiirrolla. Tietoa siirretään lohko kerrallaan siten, että niistä koostuu yksi kokonainen osio. Jokaisen lähetetyn osion jälkeen palvelin lähettää vastauspyynnön kyseiseen osioon, johon asiakkaat vastaavat joko S_COMPLETE- tai S_INCOMPLETE-vastauksella. Mikäli yksi tai useampi asiakas lähettää S_INCOMPLETE-vastauksen, lähetetään kyseinen osio uudelleen näille asiakkaille. Kun kaikki asiakkaat ovat lähettäneet S_COMPLETE-vastauksen, palvelin siirtyy lähettämään seuraavaa osiota. Kun palvelin on lähettänyt viimeisen osion datasta, lähetetään asiakkaille F_COMPLETE-ilmoitus ja siirrytään kolmanteen vaiheeseen.

Kolmannessa vaiheessa asiakkaat tarkistavat MD5-tarkistussummalla, onko palvelimen ja asiakaskoneella olevien tiedostojen tarkistussummat samat. Tieto lähetetään palvelimelle, jonka jälkeen siirrytään neljänteen vaiheeseen. Neljännessä eli istunnon katkaisuvaiheessa asiakkaat lähettävät COMPLETE-vastauksen palvelimelle, jonka jälkeen tiedonsiirto on valmis kokonaisuudessaan.



Kuva 12: ROM-datapaketin rakenne.

Kuvassa 12 on esitetty ROM-protokollan datapaketin rakenne. Paketti alkaa OP-koodilla, jolla ilmaistaan onko kyseessä varsinainen datapaketti vai vastaus palvelimelle. Paketin id:llä varmistetaan, että lähetetty paketti kuuluu tiettyyn istuntoon. Paketin id on sama kuin istunnon id. Osion ja lohkon numerot merkitsevät mitä kohtaa datasta ollaan siirtämässä. Vaihtoehtoisesti näitä kohtia käytetään ilmoittamaan palvelimelle esimerkiksi uudelleenlähetyspyynnössä siitä, mitkä kohdat ovat jääneet asiakkaalta saamatta. Datan koko vaihtelee riippuen siitä, onko kyseessä vastauspaketti vai datan siirtopaketti. Mikäli kyseinen paketti on siirtopaketti, dataosio vie jäljelle jäävän tilan paketista, kun otetaan pois edellä mainittujen otsikkotietojen koko. Muissa tapauksissa dataosio voi pitää sisällään OP-koodille merkitsevää dataa, kuten esimerkiksi virheilmoituksen.

6. Toteutuksen arviointia

Toteutuksen arviointiin käytetään kahta eri menetelmää. Ensimmäinen menetelmä on datan kopiointinopeuden mittaaminen valmiiksi asennettuihin tietokoneisiin Ohjelmistojakelijaa käyttäen ja ilman sitä. Toinen menetelmä on kokonaisasennusnopeuden mittaaminen Ohjelmistojakelijan kanssa ja ilman.

Testeissä käytettävät tietokoneet ovat mallia Dell 790, joissa on i5 3300 prosessori, 16GB keskusmuistia sekä 1GB:n verkkokortti. Käyttöjärjestelmänä on 64-bittinen Windows 7 Ultimate. Testissä käytettävä verkko on eriytetty muusta Tieturi Oy:n verkosta. Tällä varmistetaan, että testituloksiin ei vaikuta muu yleinen verkkoliikenne.

Ensimmäisessä testissä tietokoneet kopioivat itsenäisesti ilman ohjelmistojakelijaa 20.8GB dataa palvelimelta paikalliselle kovalevylle. Kopioitava data koostuu kahdesta virtuaalitietokoneen kovalevytiedostosta, jotka kummatkin ovat kooltaan 10.4GB. Testin ensimmäisessä osassa kopioivia tietokoneita on yksi, toisessa kolme ja viimeisessä osassa kuusi. Vastavuoroisesti kopiointi suoritetaan myös kolmessa osassa käyttäen hyväksi Ohjelmistojakelijaa.

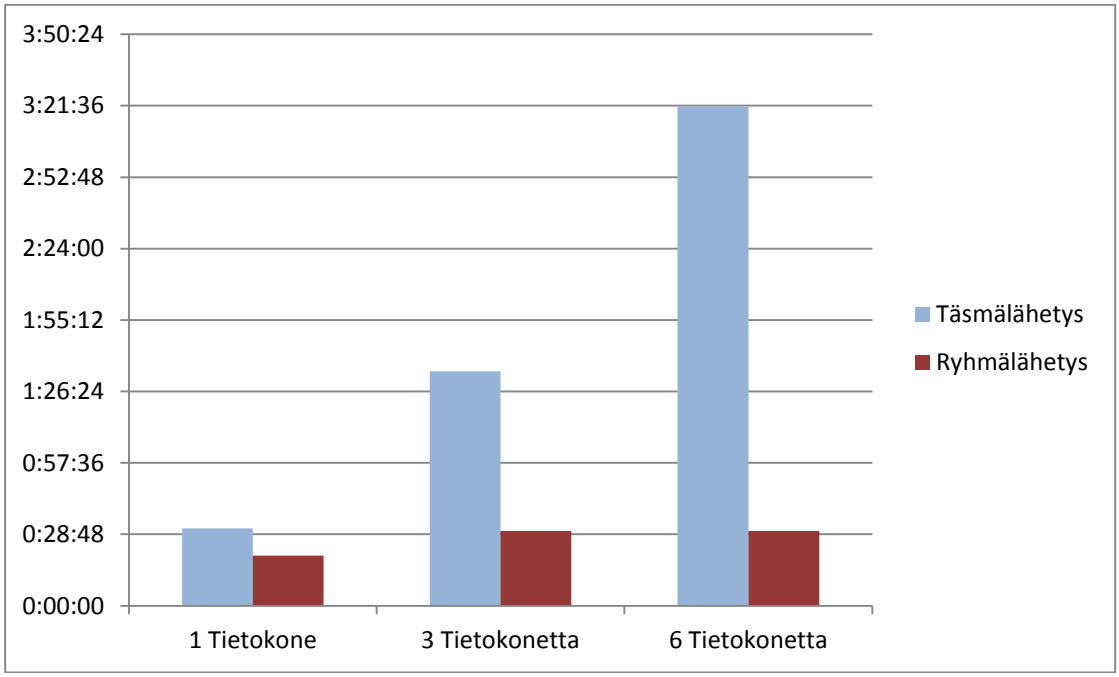
Toisessa testissä tutkitaan, minkälaisia erot ovat asennusajoissa Ohjelmistojakelijan kanssa ja ilman, kun asennetaan pieni luokkaympäristö valmiiksi ”alusta loppuun”. Testissä käytetään Windows Server 2012 -pohjalevykuvaa, johon kopioidaan Windows SQL Server 2012:sta asennusmedia. Kyseinen luokka-asennus on yleinen ja vastaa todellisuutta, lisäksi luokka-asennus on erinomaisen toimiva testissä, sillä Windows SQL Server 2012:sta ei voida suoraan asentaa pohjakuvaan ja palauttaa jokaiselle koneelle. Tämä johtuu siitä, että SQL Server muistaa asennuksen jälkeen tietokoneen nimen, jolloin mikäli koneen nimi vaihtuu palautuksen yhteydessä, SQL Server lopettaa toiminnan.

Ilman Ohjelmistojakelijaa asennusketju muodostuu pohjakuvan kopioimisesta yhdelle tietokoneelle, jonka jälkeen pohjakuvaan kopioidaan tarvittava asennusmedia. Kopioinnin jälkeen suoritetaan vaadittavat järjestelyt, jotta tietokoneesta voidaan tehdä levykuva. Järjestelyt sisältävät käyttäjätilien puhdistamisen sekä Windows Sysprep -ohjelman ajamisen. Tämän jälkeen tietokoneesta otetaan levykuva, joka tallennetaan palvelimelle. Palvelimelle tallennettu levykuva palautetaan luokassa oleville kuudelle koneelle ja levykuvan palauttamisen jälkeen tietokoneet valmistellaan käyttövalmiiksi asettamalla niille uudet nimet sekä verkkoasetukset. Tämän jälkeen jokaiselle koneelle kopioitu asennusmedia asennetaan. Asennuksen jälkeen tietokone on valmis.

Ohjelmistojakelijan myötä asennusketju yksinkertaistuu huomattavasti. Asennusketju muodostuu kopioimalla pohjakuva suoraan koko luokalle, eli kuudelle tietokoneelle. Pohjakuvan kopioimisen

jälkeen tietokoneet valmistellaan antamalla tietokoneille uusi nimi sekä oikeat verkkoasetukset. Tämän jälkeen asennettava media kopioidaan jokaiselle koneelle käyttäen Ohjelmistojakelijaa. Kun Ohjelmistojakelija on suorittanut tiedonsiirron, se automaattisesti käynnistää asennusohjelman asennusmediasta.

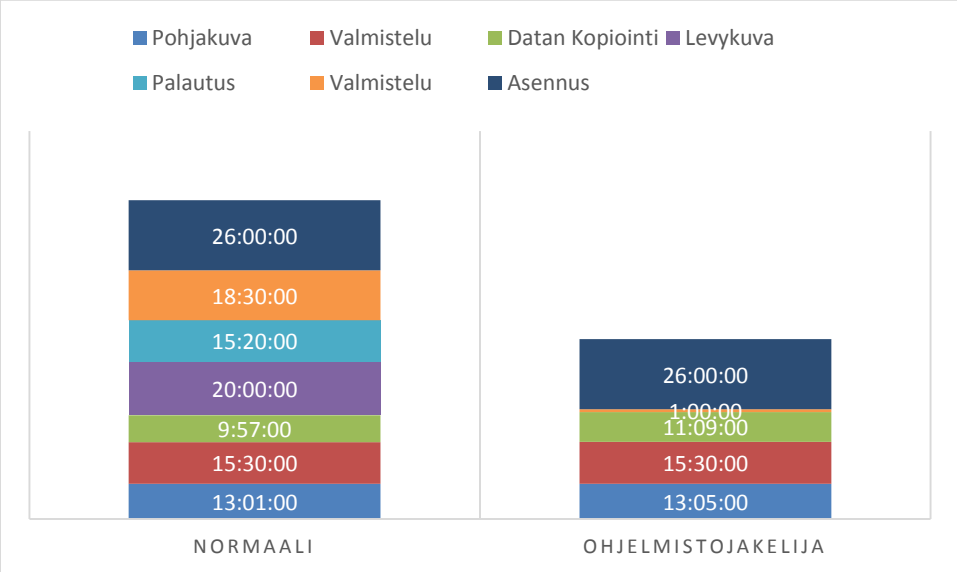
Ohjelmistojakelijalla suoritettu asennus on yksinkertaisempi kuin kloonaustyökalua käytetyssä asennuksessa, koska asennusmediaa ei tarvitse sisällyttää uuteen levykuvaan. Tällöin ei myöskään tarvitse kopioida uutta levykuvaa palvelimelle ja takaisin toisiin tietokoneisiin, jotta ryhmälähetyksestä saataisiin tarvittava hyöty.



Kuva 13: Testi 1, täsmälähetyksen ja ryhmälähetyksen vertailu.

Ensimmäisessä testissä, jonka tulokset on esitetty kuvassa 13, huomataan, miten valtava ero siirtoajoissa on täsmälähetyksen ja ryhmälähetyksen välillä. Yhden tietokoneen kopiointinopeus pysyttelee vielä kohtalaisen samana, vaikkakin ryhmälähetys on hieman nopeampi jo tässä vaiheessa. Syynä nopeuteen on ryhmälähetyksessä käytetyn protokollan yksinkertaisuus verrattuna täsmälähetykseen. Kun dataa kopioidaan kolmelle tietokoneelle, on ero jo tuntuva näiden kahden protokollan välillä. Karkeasti voidaan sanoa, että täsmälähetyksen kopiointiaika voidaan kertoa vastaanottajien määrällä, kun taas ryhmälähetyksen nopeus pysyy miltei vakiona riippumatta vastaanottajien lukumäärästä. Kyseisessä testissä ei otettu huomioon aikaa, joka kuluu Ohjelmistojakelijalta kun se pakkaa datan yhdeksi tiedostoksi. Mikäli kyseinen operaatio otetaan huomioon, on yhden tietokoneen kopiointi nopeampaa täsmälähetystä käyttäen, mutta jo kahden ja sitä useamman koneen kopiointi on nopeampaa ryhmälähetyksen avulla.

Toisen testin tulokset ovat esitetty kuvassa 14. Kuvasta käy selvästi ilmi, että Ohjelmistojakelijaa käyttämällä asennus nopeutuu verrattuna normaaliin asennustapaan. Nopeuden parantumisen ohella myös asennusketju Ohjelmistojakelijaa käyttäessä yksinkertaistuu, sillä asennusketjusta jää pois kolme vaihetta. Normaalin asennusketjun vaiheet ovat (1) pohjakuvan palautus, (2) pohjakuvan valmistelu, siten että Windows on käyttövalmis, (3) asennusmedian kopiointi tietokoneelle, normaalissa asennustavassa yhdelle tietokoneelle ja Ohjelmistojakelijan tapauksessa jokaiselle 16 tietokoneelle, (4) levykuvan ottaminen ja kopioiminen palvelimelle, (5) kohdan (4) levykuvan palautus kaikille lopuille 15 asennettavalle tietokoneelle, (6) kaikkien palautettujen tietokoneiden valmistelu siten, että Windows on käyttövalmis, ja viimeiseksi (7) halutun ohjelman asennus asennusmediasta. Ohjelmistojakelijaa käyttäessä kohdat (12) ja (13) jäävät kokonaan pois asennusketjusta sekä kohta (13) manuaalisen asentamisen sijaan on palvelimelta lähetettävä käskyn anto.



Kuva 14: Testi 2, koneluokan asennus.

Jälkimmäisessä testissä käytetyt pohjakuvat (1) ovat räätälöity kummallekin asennustavalle erikseen. Ohjelmistojakelijan käyttämässä pohjakuvassa Ohjelmistojakelijan asiakassovellus on asennettu kuvaan ennalta, jolloin Ohjelmistojakelija on toiminnassa saman tien, kun Windows on käynnissä. Kummankin pohjakuvan valmistelu (2) pitää sisällään Windowsin asennusten määrittelyn kuten esimerkiksi tietokoneen nimen ja verkkoasetukset, mutta kohdassa käytettyyn aikaan on myös laskettu mukaan asentajalta asetuksien määrittelyyn kulunut aika. Datan kopiointi (3) on normaalissa asennustavassa hivenen nopeampi kuin Ohjelmistojakelijan käyttämässä asennustavassa. Syynä tähän on, että Ohjelmistojakelijan on pakattava haluttu hakemistorakenne ennen sen lähettämistä asiakaspäätteille. Tässä kohtaa tulee kuitenkin ottaa huomioon, että

normaalissa asennustavassa data kopioituu vain yhdelle koneelle, kun Ohjelmistojakelijan tapauksessa data kopioituu viidelletoista koneelle.

Kohdat (4) ja (5) pätevät ainoastaan normaaliin asennustapaan. Kun data on kopioitu yhdelle koneelle, se valmistellaan levykuvaksi, joka siirretään palvelimelle. Käytännössä kaikki koneessa oleva data, niin haluttu asennusmedia kuin myös itse Windows, kopioidaan yhdeksi tiedostoksi palvelimen tiedostojärjestelmään. Sieltä se palautetaan ryhmälähetystä käyttäen lopuille asennettaville koneille kohdan (1) tavoin. Kohdat (4) ja (5) ovat eniten aikaa vievät kohdat asennusketjussa verrattuna Ohjelmistojakelijaan.

Kohta (6) pitää normaaliasennuksessa sisällään samat asiat kuin kohta (2), mutta lisäksi myös asennettavan sovelluksen asennuksen aloittamisen. Normaaliasennuksessa sovelluksen aloitus täytyy jokaisella koneella aloittaa manuaalisesti kirjoittamalla komentoriville sovelluksen asennustiedoston nimi sekä parametreiksi haluttu konfiguraatio. Ohjelmistojakelijan tapauksessa tämä tulee suoraan datan siirron valmistuttua keskitetysti palvelimelta. Itse sovellus asentuu (7) jokaisella koneella lokaalisti, joten asennuksen ajoissa ei ole suuria heittoja koneiden välillä.

Kuvan 14 perusteella voidaan todeta, että Ohjelmistojakelijaa käyttämällä asennus nopeutui huomattavasti, vaikka asennuksessa käytettiin dataa vain 3.5 gigabittiä. Normaalissa asennuksessa aikaa kului kaikenkaikkiaan 118 minuuttia, kun Ohjelmistojakelijan kanssa aikaa kului vain 66 minuuttia. On myös hyvä huomata, että mikäli kopioitava datamäärä kasvaa, kasvaa normaalin asennuksen aikakin, koska tiedonsiirtoa edestakaisin tapahtuu enemmän. Ohjelmistojakelijaa käyttämällä tiedon edestakaista siirtoa ei tapahdu, jolloin asennusaika pitenee vain suhteessa asennusmedian kokoon.

6.1. Sovelluksen heikkoudet

Ohjelmistojakelijan toiminnan periaatteen mukaisesti palvelinsovellus pakkaa jokaisen lähetettävän datan yhdeksi tiedostoksi. Pakkauksen ei tarvitse käyttää mitään algoritmia, mutta hakemistot ja tiedostot on saatettava yhdeksi tiedostoksi, jonka ohjelmistojakelija voi levittää asiakkaille. Tämä on niin Ohjelmistojakelijan vahvuus kuin myös tietyissä tilanteissa heikkous. Esimerkiksi tilanteessa, jossa datan määrä ei ole järin suuri mutta vastaanottajien määrä on, olisi nopeampaa lähettää tiedostot yksitellen ilman pakkausta. Tämä johtuu siitä, että datan pakkaukseen kuluu aikaa, vaikka mitään algoritmia ei käytettäisiinkään. Tämä aika voitaisiin käyttää suoraan datan siirtämiseen palvelimelta asiakkaille. Mikäli datan pakkauksesta saatu hyöty (datan koon pienentyminen) suhteessa siihen käytettyyn aikaan ei pienennä kokonaissiirtoaikaa verrattuna pakkaamattoman datan lähettämiseen, on pakotetun pakkauksen käyttö haitallista. Suurimmassa osassa tilanteita tämä ei kuitenkaan ole ongelma, sillä Ohjelmistojakelijaa käytetään lähinnä suurien data määrien siirtämiseen ja asiakkaiden määrä on hyvin rajattu.

Ohjelmistojakelijan toiseksi ongelmaksi voi muodostua tietyissä tilanteissa kopiointinopeus. Mikäli verkossa on ruuhkaa tai yhdelläkin siirtoon osallistuvasta asiakkaasta on ongelmia laitepuolen kanssa (kovalevy, verkkokortti), hidastuu kaikkien osallistujien datan siirto. Syynä tähän on ryhmälähetyksen ja sen päälle rakennettujen protokollien keskeinen toimintaperiaate, jonka mukaan data lähetään hitaimman vastaanottajan tahdilla, jotta jokainen asiakas pysyy mukana lähetyksessä. Näin ollen, vaikka yhdeksän kymmenestä koneesta pystyisi vastaanottamaan dataa sadan megatavun sekuntinopeudella, mutta yksi kymmenestä vain kymmenen megatavun sekuntinopeudella, joutuvat kaikki toimimaan hitaimman eli kymmenen megatavun sekuntivauhdilla.

Puutteeksi Ohjelmistojakelijassa on huomattu myös datan siirron jälkeisten komentojen oikeellisen ajon varmistus. Data ja datansiirron onnistuminen ja oikeellisuus voidaan varmistaa helposti käyttämällä tarkistussummaa, mutta siirron jälkeisten komentojen toimintaa on vaikea valvoa ja varmistaa. Tästä syystä ohjelman käyttäjän on varmistettava asennuksen toiminta henkilökohtaisesti, mikäli asennuksen jälkeisiä komentoja on ajettu.

6.2. Sovelluksen vahvuudet

Ohjelmistojakelijan ehdottomiin vahvuuksiin kuuluu mahdollisuus aloittaa datan siirto mistä tahansa laitteesta, jossa on HTML5-tekniikkaa tukeva web-selain. Tämän avulla kurssiasennukset voidaan aloittaa etäältä VPN-yhteyden kautta, jolloin asennus voidaan aloittaa aikaisemmin, eikä asentajan itse tarvitse odottaa siirron valmistumista työpisteellä, vaan hän voi vasta siirron valmistuttua tulla viimeistelemään asennuksen.

Vahvuuksiin voidaan lukea myös automaattinen yhteydenotto palvelinsovellukseen sekä palvelinsovelluksen kyky muistaa olemassaolevat asiakkaat. Tämän johdosta palvelinsovelluksen käyttöliittymästä voidaan valita tunnetuista asiakkaista ne, joille lähetetään dataa ilman, että asiakassovelluksesta pitää erikseen manuaalisesti ottaa yhteys palvelimeen. Näin saadaan muodostettua yksinkertainen keskitetty hallinta, jota voidaan käyttää muuhunkin kuin pelkästään tiedonsiirtoon, esimerkiksi tietokoneiden etäkäynnistykseen ja asetuksien vaihtoon.

7. Yhteenveto ja kehittämisehdotukset

Ohjelmistojakelijan perimmäinen tarkoitus on nopeuttaa tapaa, jolla nykyisin asennetaan kursseja, jotka vaativat suuria määriä dataa kopioitavaksi jokaiselle koneelle. Tästä tehtävästä Ohjelmistojakelija suoriutuu erinomaisesti. Testeistä saatujen tulosten perusteella jo kohtalaisen pienenkin datamäärän (20 GT) kopiointi usealle koneelle (kuusi tietokonetta) nopeutuu huomattavasti, puhumattakaan kun kyseessä on useampi kone ja suurempi datamäärä.

Kun asennuksiin kuluvaa aikaa saadaan pienennettyä, ei siitä ole pelkästään hyötyä yrityksen IT-henkilökunnalle työn pienentymisen muodossa, vaan parhaassa mahdollisessa tapauksessa voidaan viikkoon mahduttaa useampi vaativampi kurssi luokkaa kohden, kun vaativille kursseille ei välttämättä tarvitse varata enää yhtä tai useampaa päivää asennuksia varten.

Koska ohjelmistojakelija on suunniteltu Tieturi Oy:ssa ja sen IT-infrastruktuuria hyväksikäyttäen, on hyvin todennäköistä, että samankaltaista hyötyä ei saataisi aikaan muissa ympäristöissä. Ohjelmistojakelija itsessään vaatii nopean palvelimen sekä verkon, joka tukee ryhmälähetystä, jotta Ohjelmistojakelijasta saataisiin suurin mahdollinen hyöty irti. Myös asennettavat koneet tulisi olla standardoituja sekä eriytetty tuotantoverkosta.

Ohjelmistojakelijaa on mahdollista jatkokehittää usealla tavalla. Tärkein kehittämisalue on pakkaukseen liittyvät seikat sekä sen optimoiminen. Esimerkiksi pakkauksen sisällyttäminen suoraan tiedonsiirtoon voisi nopeuttaa datansiirtoa. Kun pakkaus suoritettaisiin siten, että pakattu data lähetettäisiin suoraan datansiirtona, eikä dataa kirjoitettaisiin ensin tiedostoon, voitaisiin pakkaus suorittaa samanaikaisesti datansiirron kanssa. Tämä tosin saattaisi hidastaa siirrettävän datan nopeutta riippuen lähettävän palvelimen kovalevyn lukunopeudesta ja verkon nopeudesta.

Toinen jatkokehitysmahdollisuus on parantaa asennuksien onnistumisen varmistamista liittyen asennusten jälkeisiin komentoihin. Ohjelmistojakelijaan voitaisiin lisätä sisäinen inventaario, jota asiakassovellukset keräisivät. Tämän avulla voitaisiin tutkia, onko jokin sovellus asennettu onnistuneesti koneeseen tai onko esimerkiksi Windowsin systeemirekisterissä jälkikomennoissa määritellyt muutokset tulleet voimaan.

Viiteluettelo

Blizzard. (2013). *Firewall, Proxy, Router and Ports Configuration for Blizzard Games*. Available from https://eu.battle.net/support/en/article/firewall-configuration-for-blizzard-games

Cerf, V. & Kahn, R. (1974). A Protocol for Packet Network Intercommunication. *IEEE Transactions on Communications* 22 (5): 637–648.

Cisco. (2012). *X.25*. Available from http://docwiki.cisco.com/wiki/X.25

Deering, S. & Hinden, R. (1998). *Internet Protocol, Version 6 (IPv6) Specification*, Available from http://tools.ietf.org/html/rfc2460

Digia. (2013). Available from http://qt.digia.com/About-us/

Federal Standard 1037C. (1996). *Telecommunications: Glossary of Telecommunication Terms*. Available from http://www.its.bldrdoc.gov/fs-1037/fs-1037c.htm

Gallaugher, J. & Ramanathan, S. (1996). *The Critical Choice of Client Server Architecture: A Comparison of Two and Three Tier Systems, Information Systems Management.* New York, Auerbach Publications. Available from https://www2.bc.edu/~gallaugh/research/ism95/cccsa.html

Gamma, E., Helm, R., Johnson, R. & Vlissides, J. (1994). *Design Patterns: Elements of Reusable Object-Oriented Software*.

GNU.org. (2013). *The GNU Operating System*. Available from http://www.gnu.org/gnu/gnu.html

GZIP.org. (2003). Available from http://www.gzip.org/

IEEE Reliability Society. (2013). Available from http://rs.ieee.org/

IEEE. (2012). *Internet Awards*. Available from http://www.ieee.org/documents/internet_rl.pdf

IETF (1). (1981). *IPv4 TRANSMISSION CONTROL PROTOCOL*. Available from http://tools.ietf.org/html/rfc7931

Ihalainen, P. (2011). *WSS – WebSocketServer*. Available from https://bitbucket.org/merc/wss-websocketserver

ISO/IEC, 9241-11. (1998). *Ergonomic requirements for office work with visual display terminals (VDT)s - Part 11 Guidance on usability. 1998: ISO/IEC 9241-11: 1998 (E).*

Lewis, P. (1995). *Peering Out a 'Real Time' Window*. The New York Times. February 08, 1995.

Liu, G. (2011). *gziptest.sh part 2: multi-threaded compression benchmarks*. Available from http://vbtechsupport.com/1614/2/

Microsoft MSDN. (2010). *Internet Protocol version 4 Address Classes.* Available from http://msdn.microsoft.com/en-us/library/aa918342.aspx

Microsoft SCCM. (2013). *About Distribution Points*. Available from http://technet.microsoft.com/en-us/library/bb680614.aspx

Microsoft. (2012). *Flexible Workstyles and Enterprise IT, Supporting the Consumerization of IT with an Intelligent Infrastructure*. Available from http://download.microsoft.com/download/9/6/A/96A4E54E-C09F-44EF-BBF6-AAFD4721C929/Intelligent Infrastructure White Paper.pdf

Microsoft. (2013). *System Center 2012 R2*. Available from http://www.microsoft.com/en-us/server-cloud/system-center/configuration-manager-2012.aspx

OMG.org. (2005). *Documents associated with UML Version 2.0*. Available from http://www.omg.org/spec/UML/2.0/

Postel, J. (1980). *RFC 768: User Datagram Protocol*. Available from http://tools.ietf.org/html/rfc768

RFC 1350. (1992). *The TFTP Protocol(Revision 2).* Available from http://tools.ietf.org/html/rfc1350

RFC 2009. (1996). *GPS-Based Addressing and Routing.* Available from http://tools.ietf.org/html/rfc2009

Savetz, K., Randall, N. & Lepage, Y. (1996). *MBONE: Multicasting Tomorrow's Internet*. Available from http://www.savetz.com/mbone/

Symantec, (2013). *Welcome to Ghost.com.* Available from http://www.symantec.com/themes/theme.jsp?themeid=ghost

Symantec. (2007). *How Ghost Multicasting communicates over the network.* Available from http://www.symantec.com/business/support/index?page=content&id=TECH106806

Symantec. (2009) *Ghost compression ratios and performance*. Available from http://www.symantec.com/docs/TECH106775

Symantec. (2013). *Symantec Ghost Solution Suite*. Available from http://www.symantec.com/ghost-solution-suite

Thyagarajan, A., Casner S. & Deering, S. (1995). *Paper: Making the MBone Real*. Available from http://www.isoc.org/inet95/proceedings/PAPER/227/html/paper.html

Tieturi Oy. (2013). Haettu osoitteesta http://www.tieturi.fi

Wikipedia. (2013). *McMurdo Station*. Available from http://en.wikipedia.org/wiki/McMurdo_Station

Zytrax. (2013). *Appendix D: DNS and Relevant RFCs*. Available from http://www.zytrax.com/books/dns/apd/