Pasi Lahti

Hajasijoitettujen päätelaitteiden ohjelmistojen etähallintaratkaisu

Elektroniikan, tietoliikenteen ja automaation tiedekunta

Diplomityö, joka on jätetty opinnäytteenä tarkastettavaksi diplomi-insinöörin tutkintoa varten Espoossa 19.4.2010.

Työn valvoja:

Prof. Jukka Manner

Työn ohjaaja:

SM Kaius Häggblom

Aalto-yliopisto
Teknillinen korkeakoulu

AALTO-YLIOPISTO
TEKNILLINEN KORKEAKOULU

DIPLOMITYÖN
TIIVISTELMÄ

Tekijä: Pasi Lahti

Työn nimi: Hajasijoitettujen päätelaitteiden ohjelmistojen etähallintaratkaisu

Päivämäärä: 19.4.2010 Kieli: Suomi Sivumäärä:8+60

Elektroniikan, tietoliikenteen ja automaation tiedekunta

Tietoliikenne- ja tietoverkkotekniikan laitos

Professuuri: Tietoverkkotekniikka Koodi: S-38

Valvoja: Prof. Jukka Manner

Ohjaaja: SM Kaius Häggblom

Tämä diplomityö tutkii mahdollisuutta muodostaa etähuoltoyhteys hajasijoitetuille M2M päätelaitteille. Huoltoyhteyttä ei voi muodostaa suoraan, koska päätelaitteet on kytketty tietoverkkoihin, joissa käytetään osoitteenmuunnosta tai palomuureja. Tästä syystä työssä kehitettiin ohjelmistokomponentteja, joilla on mahdollista käskeä huoltoyhteys auki palvelimelta, mutta huoltoyhteyden muodostaa päätelaite. Huoltoyhteys toteutettiin käyttämällä apuna SSH-ohjelmiston tarjoamaa käänteistä tunnelia. SSH ratkaisi osoitteenmuunnosongelman mutta ei tehoa palomuureihin. Palomuurien kiertämiseksi toteutettiin muutos päätelaitteen yhteystapalogiikkaan, jolla päätelaitteen voi pakottaa käyttämään GPRS-yhteyttä, koska GPRS-verkoissa ei ole palomuureja esteenä. Testauksessa päivitettiin luodun huoltoyhteyden avulla Linux-ydin ja ajurimoduulit päätelaitteille, jotka oli sijoitettu kahdeksaan eri maahan ja neljään eri tyyppiseen kohteeseen. Testin tuloksena kaikki päätelaitteet saatiin päivitettyä.

Avainsanat: M2M, NAT, palomuuri,
SSH, GPRS, Ethernet

AALTO UNIVERSITY
SCHOOL OF SCIENCE AND TECHNOLOGY

ABSTRACT OF THE
MASTER'S THESIS

Author: Pasi Lahti

Title: Remote Management of Software in Distributed Terminal Devices

Date: 19.4.2010 Language: Finnish Number of pages:8+60

Faculty of Electronics, Communications and Automation

Department of Communications and Networking

Professorship: Networking Technology Code: S-38

Supervisor: Prof. Jukka Manner

Instructor: SM Kaius Häggblom

This Master's Thesis explores possibility to establish a service connection to distributed M2M terminal devices. The service connection can not be established directly because terminal devices are connected to networks which use network address translation or firewalls. For this reason, new software components were developed to enable the service connection in a way that command is given from the server but actual connection is established by the terminal device. The service connection was implemented using reverse tunnel provided by SSH software. SSH solved the network address translation problem but not the firewall problem. To bypass firewalls, the current measurement software was modified in a way that it is possible to force the terminal device to use GPRS connection instead of Ethernet because GPRS network has no firewalls. In testing phase, a Linux kernel and driver modules in terminal devices was updated using the service connection. Terminal devices were located in eight different countries and were installed on four different types of machinery. As a result, all the terminal devices succeeded to update.

Keywords: M2M, NAT, Firewall,
SSH, GPRS, Ethernet

Esipuhe

Haluan kiittää työn ohjaajaa Kaius Häggblomia diplomityön aiheen keksimisestä ja ideoinnista sekä lisäksi Oliotalon M2M laitteiston ja tilojen tarjoamisesta diplomityön vaatiman ohjelmistokehityksen ajaksi. Kiitos kuuluu myös Oliotalon urheille asiakkaille, joiden päätelaitteita tässä diplomityössä käytettiin testauksen uhrina.

Haluan kiittää myös työn valvojaa Jukka Mannerta diplomityötä koskevista rakentavista kommenteista.

Erityisesti haluan kiittää rakasta vaimoani Matleenaa, joka on jaksanut katsella miestään useamman kuukauden ajan käyttämässä lukuisia tunteja tietokoneen ääressä diplomityötä kirjoittaen.

Suuri kiitos kuuluu myös rakkaille vanhemmilleni Asserille ja Irmalle, jotka ovat tukeneet opintojani.

Edellä mainitut henkilöt ovat mahdollistaneet sen, että tämä työ on vihdoinkin valmis.

Kivenlahti, 19.4.2010

Pasi Lahti

Sisältö

Symbolit ja lyhenteet

Lyhenteet

M2M	Machine-to-Machine eli koneviestintä. Vähintään kaksi konetta viestii tietoverkon (yleensä Internet) välityksellä ilman ihmisen vuorovaikutusta.
RFID	Radio Frequency IDentification on tunnisteen etälukeminen/ -kirjoittaminen käyttäen radiotaajuuksia. Käyttökohde on esimerkiksi kulkukortit, joilla voidaan availla ovia.
GPS	Global Positioning System on maailmanlaajuinen satelliittipaikannusjärjestelmä, jonka on kehittänyt ja rahoittanut Yhdysvaltain puolustusministeriö palvelemaan sotilastoimintaa. Järjestelmää voidaan nykyään käyttää myös siviilikäytössä. Suosituin käyttökohde on navigaattorit.
GPRS	General Packet Radio Service on pakettikytkentäinen tiedonsiirtopalvelu GSM verkossa.
GSM	Global System for Mobile communication on maailmanlaajuinen matkapuhelinjärjestelmä.
SSH	Secure SHell on verkkoprotokolla, joka tarjoaa salatun tiedonsiirtokanavan kahden laitteen välille. SSH korvaa salaamattoman Telnet protokollan.
NAT	Network Address Translation on Internetissä käytettävä osoitteenmuunnostekniikka, jolla julkisen IP-osoitteen taakse voidaan muodostaa uusi yksityinen verkko, jonka kaikki laitteet esiintyvät samalla julkisella IP-osoitteella Internetiin päin. Tällä tekniikalla säästetään julkisia IP-osoitteita.
SHM	Structural Health Monitoring on rakenteiden kunnonvalvontaa, jonka avulla pyritään tunnistamaan vaurioita.
FFT	Fast Fourier Transform eli nopea Fourier-muunnos on tehokas algoritmi DFT:n laskentaan tietokoneella.
DFT	Discrete Fourier Transform eli diskreetti Fourier-muunnos on Fourier-muunnos, jolla digitaalisesti näytteistetty aikatason signaali voidaan muuttaa taajuustason spektriksi.
UMTS	Universal Mobile Telecommunications System on kolmannen sukupolven matkapuhelinteknologia, jonka suunnittelussa on otettu huomioon myös nopea datasiirto.
WLAN	Wireless Local Area Network on langaton lähiverkko, jossa voidaan liikennöidä IP-paketeilla.
IP	Internet Protocol on Internetissä käytettävä pakettimuotoinen liikennöintitapa.

SMS	Short Messaging Service on matkapuhelinverkon tekstiviesti-järjestelmä, jossa voidaan lähettää 160 merkin mittaisia viestejä laitteesta toiseen.
USB	Universal Serial Bus on sarjaväyläarkkitehtuuri oheislaitteiden liittämiseksi tietokoneeseen.
CAN	Controller Area Network on tyypillisesti ajoneuvoissa käytettävä automaatioväylä. Väylällä liikkuu tietoa ajoneuvossa tapahtuvista toiminteista.
VGA	Video Graphics Array on standardi, jolla kuva välitetään näyttölaitteelle.
LED	Light-Emitting Diode on puolijohdekomponentti, joka säteilee valoa.
GNU	GNU's Not Unix projekti on vapaiden ohjelmistojen alullepanija.
JFFS2	Journaling Flash File System on flash-muistin päällä toimivan tiedostojärjestelmän toinen versio.
UBIFS	Unsorted Block Image File System on flash-muistin päällä toimiva tiedostojärjestelmä kuten JFFS2 mutta kehittyneempi.
TCP	Transmission Control Protocol on tietoliikenneprotokolla, jonka välityksellä voidaan siirtää tavuja kahden koneen välillä luotettavasti.
EJB	Enterprise JavaBeans määrittää, miten tietoa tulee hallita ja tallentaa.
JSP	JavaServer Pages on palvelinpuolen teknologiaa, jonka avulla on mahdollista generoida dynaamista sisältöä verkkosivulle.
SQL	Structured Query Language on standardoitu kyselykieli, jolla voidaan tehdä hakuja, lisäyksiä, poistoja ja muutoksia relaatiotietokantaan.
SMTP	Simple Mail Transport Protocol on TCP-pohjainen protokolla, jolla voidaan välittää sähköpostiviestejä.
WWW	World Wide Web on Internetissä toimiva maailmanlaajuinen hajautettu hypertekstijärjestelmä.
HTTP	HyperText Transfer Protocol on protokolla, jolla selaimet hakevat tietoa WWW-palvelimelta.
HTML	HyperText Markup Language on avoimesti standardoitu kuvauskieli, jolla voidaan kuvata hyperlinkkejä sisältävää tekstiä.
OSI	Open Systems Interconnection jaottelee verkon eri protokollat ja palvelut seitsemään eri kerrokseen, jossa ylemmät kerrokset käyttävät alempia hyväkseen.

FTP	File Transport Protocol on TCP-protokollaa käyttävä menetelmä tiedostojen siirtämiseen kahden koneen välillä.
URL	Uniform Resource Locator on osoite, joka osoittaa verkkosivun sijainnin WWW:ssä.
DNS	Domain Name System on Internetin nimipalvelujärjestelmä, jolla selkokielinen osoite muunnetaan numeeriseksi IP-osoitteeksi.
MIME	Multipurpose Internet Mail Extensions on tapa määritellä sähköpostiviestin sisältö. MIME:n avulla voidaan välittää sähköpostiviestissä muutakin sisältöä kuin tekstiä.
SIM	Subscriber Identity Module on älykortti, johon on tallennettu matkapuhelinverkon tilaajaan yksilöllinen IMSI-koodi.
IMSI	International Mobile Subscriber Identity on SIM-kortille tallennettu numerosarja, joka identifioi matkapuhelin-verkon käyttäjän.
PAT	Port Address Translation on yksi osoitteenmuunnoksen tyyppi, jossa ulkoverkosta voidaan ohjata portin perusteella sisäverkkoon liikennettä tietylle koneelle tiettyyn porttiin.
NAPT	Network Address Port Translation on toinen nimi PAT:lle.
RSA	Epäsymmetrinen julkien avaimen salausalgoritmi, jonka kirjaimet tulevat keksijöiden sukunimistä Rivest, Shamir ja Adleman.
MD5	Message Digest on algoritmi, jolla lasketaan tavujonosta tiiviste, joka on 128-bittinen.
DHCP	Dynamic Host Configuration Protocol on protokolla, jolla verkkoon liittyville laitteille voidaan jakaa IP-osoitteet automaattisesti.

1 Johdanto

Koneviestintä on nousemassa yhä suurempaan rooliin rakenteiden ja laitteiden etävalvonnassa. Koneviestintä tulee englanninkielisestä termistä Machine-to-Machine, josta käytetään yleisesti lyhennettä M2M. Etävalvonta voidaan jakaa karkeasti kahteen kategoriaan: kunnonvalvonta ja käytönvalvonta.

Kunnonvalvonta perustuu erilaisten suureiden mittaamiseen monilta eri antureilta ympäri valvottavaa rakennetta tai laitetta. Tällaisia suureita voi olla esimerkiksi tärinä (kiihtyvyys/poikkeutus), lämpötila, öljyn laatu, kierrosnopeus ja paine. Tällaisten suureiden seuraamisesta voidaan muodostaa kuva valvottavan kohteen kunnosta ja huollon tarpeesta. Kunnonvalvonnan tärkein sovellus on huomata mahdolliset viat jo ennakoivasti, jotta uusia osia ehditään tilata ja vaihtaa ennen kuin rakenne tai laite hajoaa lopullisesti ja on mahdollisesti pois käytöstä pitkiäkin aikoja. Valvottavat kohteet ovat yleensä kooltaan suuria ja hinnaltaan kalliita, joten ennakoivalla toiminnalla voidaan saavuttaa suuri rahallinen säästö.

Käytönvalvonta perustuu myös erilaisten suureiden mittaamiseen mutta ei välttämättä antureilta. Mitattavia suureita voi olla moottorin käyttöaika (digitaalinen päällä/pois), kuljettajantunnistus (RFID), törmäykset (kiihtyvyys/poikkeutus) ja sijainti (GPS). Näiden suureiden avulla voidaan seurata laitteen käyttötunteja, kuka on käyttänyt, missä laitetta on käytetty, onko laitteella mahdollisesti törmäilty vai onko se muuten ollut rankassa käytössä. Näiden suureiden avulla voidaan myös kartoittaa laitteen huoltotarpeita ennakoivasti, jotta uusia osia voidaan tilata jo etukäteen. Laitteiden vuokrausliiketoiminnassa on mahdollista seurata, noudattaako asiakas vuokrasopimusta vai tapahtuuko rikkeitä esimerkiksi sovittujen käyttötuntien ylittämisessä. Valvottavat kohteet voivat olla liikkuvia ajoneuvoja tai paikallaan pysyviä työkoneita.

Kummallekin kategorialle yhteinen ongelma on se, että valvottava kohde on jollain tavalla hankala valvottavaksi normaalein keinoin. Normaaleilla keinoilla tarkoitetaan tässä yhteydessä sitä, että anturoidaan kohde ja yhdistetään kaapelit valvomoon, jossa kohdetta voidaan seurata monitorista. Tämä ei ole mahdollista, mikäli valvottava kohde on liikkuva ajoneuvo kuten trukki tai kuorma-auto. Toisaalta taas kohde voi olla paikallaan kuten tuulimylly tai torninosturi mutta niin hajasijoitettu, että kaapelointi on käytännössä kannattamatonta. Torninosturikaan ei ole ikuisuuksia paikallaan vaan siirtyy työmaalta toiselle, joten kaapelointi ei tässä tapauksessa tule edes kysymykseen.

Ratkaisun ongelmaan tuo M2M-viestintä. M2M-ratkaisun ansiosta valvontatieto on mahdollista siirtää kohteesta Internetissä sijaitsevalle palvelimelle. Valvontatieto siirretään yleensä langattomasti käyttäen GPRS-verkkoa, jolloin minkäänlaista kaapelointia ei tarvita. Joissakin tuulipuistoissa on kuitenkin mahdollisuus liittyä langallisesti Internetiin Ethernetin välityksellä. Tiedonkeruun ja lähettämisen hoitaa valvottavaan kohteeseen asennettava päätelaite, joka on pieni sulautettu tietokone varustettuna tarpeellisilla ominaisuuksilla ja ohjelmistoilla. Kun tieto on siirtynyt päätelaitteelta palvelimelle, voidaan palvelimella olevaa tietoa tutkia yleisesti käytössä olevilla Internet-selaimilla, eikä täten tarvitse käydä lähelläkään rakennetta tai laitetta, ellei siihen synny tarvetta. Palvelimelta voidaan konfiguroida päätelaitteel-

le erilaisia antureiden lukemiseen liittyviä konfiguraatiota ja asettaa hälytysrajoja. Sopivasti asetettujen hälytysrajojen ansiosta valvontatietoa ei välttämättä tarvitse vahtia palvelimella manuaalisesti päivittäin, vaan hälytykset voidaan ohjata joko sähköpostiin tai matkapuhelimeen tekstiviestinä. Hälytyksen sattuessa voidaan käydä katsomassa palvelimella olevaa valvontahistoriaa ja tutkia hälytykseen johtaneita syitä, minkä perusteella voidaan tehdä operatiivinen päätös jatkotoimenpiteistä.

Tämä kuulostaa helpolta ja yksinkertaiselta, mutta asia ei ole kuitenkaan loppuen lopuksi niin yksinkertainen, miltä se näyttää, koska monesti nälkä kasvaa syödessä, ja jo kentällä oleviin päätelaitteisiin vaaditaan jatkuvasti uusia ominaisuuksia. Uudet ominaisuudet puolestaan synnyttävät päivitystarpeen itse päätelaitteisiin, jotka on asennettu hankaliin paikkoihin, ja joiden tarkoitus on juurikin vähentää käyntejä hankalissa paikoissa. Uudet ominaisuudet on toteutettavissa päätelaitteen ohjelmistoon, joten rautaa eli päätelaitetta ei tarvitse useastikaan vaihtaa. Kun uusi ohjelmistoversio on saatu kehitettyä valmiiksi ja testattu toimivaksi, ei ole mielekästä lähteä asentamaan sitä ympäri maailmaa hankaliin paikkoihin. Toisen ongelman muodostaa päätelaitteen tai siihen kytkettyjen antureiden vikaantuminen. Näitäkään vikoja ei ole mielekästä lähteä metsästämään paikan päälle, jos ongelmaan on olemassa helpompi ratkaisu.

Osittain ohjelmistonpäivitysongelmaan onkin olemassa ratkaisu. Uusi ohjelmisto on mahdollista siirtää päätelaitteelle, koska päätelaite on liitetty Internetiin ja pyrkii olemaan jatkuvasti yhteydessä palvelimeen. Tällä tavoin voidaan päivittää turvallisesti vain osa ohjelmistoa. Haasteellisempi operaatio onkin päivittää koko päätelaitteen käyttöjärjestelmä. Käyttöjärjestelmän päivitys on mahdollista suorittaa automaattisesti, mutta se on sen verran riskialtis operaatio, että se halutaan ainakin alkuvaiheessa suorittaa manuaalisesti ja valvotusti. Toiseen eli vikaantumisongelmaankin ratkaisuna voitaisiin tehdä paljon ylimääräistä vikadiagnostiikkaa havainnoimaan erilaisia vikatilanteita, mutta vikoja esiintyy kuitenkin niin harvakseltaan, että ei ole järkevää sisällyttää liikaa ohjelmistoa valvomaan tilanteita, joita ei tapahdu kovin useasti.

Tämän työn tavoitteena on suunnitella ja toteuttaa sellainen huoltoyhteys, jolla yllämainitut haasteet saadaan ratkottua. Ratkaisun pohjana tullaan käyttämään käänteistä SSH-tunnelointia, joka tarjoaa salatun komentokehoteyhteyden Linux-käyttöjärjestelmällä varustettuun päätelaitteeseen. Pääongelma muodostuu siitä, että päätelaitteisiin ei voida muodostaa suoraan yhteyttä, koska ne sijaitsevat poikkeuksetta NAT:n tai palomuurin takana. Täten ainoaksi mahdollisuudeksi jää se vaihtoehto, että päätelaite muodostaa itse huoltoyhteyden palvelimeen päin. Huoltoyhteyden avaus halutaan kuitenkin tehdä keskitetysti palvelimelta käsin, joten ratkaistavaksi haasteeksi jää tämän yhteispelin toteuttaminen. Osaongelmaksi voidaan luonnehtia huoltoyhteyden onnistumisen takaaminen päätelaitteessa, vaikka Internet-yhteys olisikin hidas ja epäluotettava. Käytännössä työ on ohjelmistokomponenttien toteuttamista sekä päätelaite- että palvelinohjelmistoihin.

Työn aikana kehitettiin ratkaisu, jolla onnistuttiin luomaan huoltoyhteys eri puolilla maapalloa sijaitseviin päätelaitteisiin. Päätelaitteisiin onnistuttiin päivittämään uusi versio Linux-käyttöjärjestelmän ytimestä, minkä jälkeen päätelaite kykeni jatkamaan jälleen normaalia mittausrutiiniaan. Kehitetty ratkaisu toimii sekä GPRS-

että Ethernet-yhteydellä, jolloin useissa Ethernet-pohjaisissa lähiverkkoratkaisuissa voidaan ohittaa NAT ja palomuuri vaihtamalla yhteystapa GPRS:ksi.

Työ tehtiin Oliotalo Oy:lle osana oman M2M-alustan kehitystä. Työn aluksi lukijalle selvennetään yleisesti M2M käsitettä, ja miten M2M nivoutuu yhteen rakenteiden kunnon- ja käytönvalvonnan kanssa. Sen jälkeen esitellään Oliotalo Oy:n olemassa oleva M2M-alusta ja siinä käytettävät teknologiat, minkä jälkeen tehdään katsaus työn kannalta ongelmallisiin Internetissä käytettäviin teknologioihin, jotka mainittiin aiemmin. Tämän jälkeen luodaan lyhyt katsaus SSH:n tarjoamaan teknologiaan, ja millä tavoin se sopii tämän työn osaratkaisuksi. Näiden teoreettisten pohjustuksien jälkeen voidaan lähteä kehittelemään ratkaisua itse ongelmaan. Työn lopuksi tarkastellaan ratkaisun toimivuutta, ja pohditaan sekä sen onnistumista että mahdollisia jatkokehitysmahdollisuuksia.

2 Johdatus kunnonvalvontaan ja koneviestintään

Ennen kuin lähdetään tutkimaan työn pääongelmaa, on tarpeellista tietää perusasioita rakenteiden kunnonvalvonnasta. Termi rakenteiden kunnonvalvonta on peräisin englanninkielisistä sanoista Structural Health Monitoring eli lyhyesti SHM. Termi ei esiinny juurikaan suomenkielisessä kirjallisuudessa, koska aiheesta ei ole kirjoitettu yleispäteviä teoksia, vaan kaikki teokset keskittyvät jonkin tietyn osa-alueen kunnonvalvontaan, joten siksi tässä työssä otetaan huomioon myös englanninkieliset termit käsitteiden yhteydessä.

Tämän luvun tarkoitus on määritellä, mitä kunnonvalvonta on, minkä jälkeen esitellään eri lähestymistapoja kunnonvalvontaan. Luvun tarkoitus on myös motivoida lukijalle, miksi kunnonvalvonta oikeastaan on tarpeellista. Kun lukijalla on selkeä käsitys kunnonvalvonnasta, voidaan siirtyä tarkastelemaan, millä tavoin M2M-teknologia auttaa kunnonvalvonnan haasteissa.

2.1 Mitä kunnonvalvonta on?

Määritelmän mukaan kunnonvalvonta on prosessi vauriontunnistamisstrategian toteuttamiseksi avaruus-, siviili- ja konetekniikan infrastruktuureihin [1, s. 303]. Tässä yhteydessä vauriolla tarkoitetaan järjestelmässä esiintyviä muutoksia, jotka vaikuttavat epäedullisesti sekä nykyiseen että tulevaisuuden suorituskykyyn [1, s. 303]. Jotta vaurio voitaisiin määritelmänsä mukaan tunnistaa, tiedossa tulee olla myös sellainen tila, joka ei täytä vauriotilan tunnusmerkkejä. Toisin sanoen tiedossa pitää olla vaurioitumaton normaalitila, jotta järjestelmästä saatavia tunnusmerkkejä voidaan verrata tähän normaaliin tilaan, ja sen perusteella päättää, onko kyseessä vaurio vai ei.

Tässä työssä vauriontunnistus voidaan rajata rakenteisiin ja mekaanisiin järjestelmiin. Esimerkkejä tällaisista järjestelmistä ovat tuulimyllyn pyörivä vaihteisto, raiteilla kulkeva junanvaunu tai silta. Tällöin vaurio määritellään siten, että kyseessä on muutos materiaalissa ja/tai rakenteen geometrisissa ominaisuuksissa. Koska kyseessä on mekaaninen rakenne, vaurio ei synny välttämättä hetkessä vaan pitkän aikavälin tapahtumasarjana. Aluksi materiaalissa voi alkaa esiintyä kulumaa, jota voidaan kutsua vajaukseksi tai puutteeksi materiaalissa. Kuluma voi esiintyä aluksi vain yhdessä järjestelmän komponentissa, mutta ajan kuluessa vajauksesta johtuva tärinä tai lämpötilan nousu voi johtaa toisen lähellä olevan komponentin vaurioitumiseen. Tässä vaiheessa järjestelmä voi vielä pysyä toimintakykyisenä, mutta sen optimaalinen suorituskyky on voinut heikentyä. Koska vaurio ei itsestään häviä järjestelmästä, vaan se pikemminkin pyrkii kasvamaan, järjestelmä saavuttaa aikanaan sellaisen pisteen, jossa se lakkaa toimimasta kokonaan. Tällaista tilannetta kutsutaan vikaantumiseksi. [1, s. 304]

Vikaantumisaika voi olla joko pitkä tai lyhyt. Pitkän aikavälin tilanne voi olla yllä kuvattu kuluma materiaalissa tai vaihtoehtoisesti materiaali voi ruostua, ja pitkän aikavälin jälkeen jokin osa pettää siitä syystä, että ruosteinen kohta murtuu rikki rasituksesta. Lyhyen aikavälin vikaantuminen voi tapahtua joko aikataulun mukaisesti tai arvaamattomasti. Aikataulun mukaisesti vikaantuminen voi tapahtua esimerkik-

si silloin, kun lentokone laskeutuu lentokentälle ja juuri sillä hetkellä, kun koneen renkaat koskettavat maata, jokin osa vikaantuu, ja se johtaa lentokoneen toimintakyvyn häviämiseen. Arvaamattomia vikaantumisia voi syntyä esimerkiksi silloin, kun armeijan ajoneuvo joutuu vihollisen tulitukseen tai tapahtuu luonnonilmiö kuten maanjäristys. Tulituksessa luodit voivat tehdä pahoja materiaalimuutoksia moottoriin, jolloin ajoneuvon moottori ei enää toimi ja kyseessä on vikaantuminen. Samoin maanjäristys voi aiheuttaa siltaan sellaisia materiaalimuutoksia, että silta joko katkeaa tai vääntyy siten, että ajoneuvot eivät voi enää käyttää sitä, ja jälleen kyseessä on vikaantuminen. [1, s. 304]

Palataan takaisin kunnonvalvonnan määritelmään, jossa todettiin, että kunnonvalvonta tarkoittaa lyhyesti ilmaistuna vaurioiden tunnistamista järjestelmästä. Mietitään seuraavaksi, miten tähän tavoitteeseen päästään. Jotta järjestelmästä saataisiin tunnusmerkkejä vaurioista, sitä pitää valvoa jatkuvasti. Tämä onnistuu ottamalla järjestelmästä näytteitä tasaisin väliajoin ja tallentamalla järjestelmän tila kullakin ajanhetkellä. Näistä näytteistä voidaan etsiä suoraan merkkejä vauriosta tai vaihtoehtoisesti tehdä tilastollista analyysiä järjestelmän kunnosta. Lyhyen aikavälin muutokset, kuten maanjäristys, voidaan havaita suoraan, koska muutos edelliseen tilaan verrattuna on sen verran merkittävä, että tapahtuma ei jääne epäselväksi. Pitkän aikavälin vikaantumista, kuten osien kulumista, ei voida todeta yhdestä tilapäivityksestä, vaan avuksi tarvitaan tilastollista analyysiä, josta voidaan nähdä järjestelmän kunnon kehitys pidemmällä aikavälillä, kun vertaillaan näytteitä ja niistä syntyvää tilahistoriaa pitkälle menneisyyteen asti. [1, s. 304]

Vauriontunnistukseen on viisi hyvin lähellä toisiaan olevaa toimintatapaa, joita ovat rakenteiden kunnonvalvonta (SHM, engl. Structural Health Monitoring), kunnonvalvonta (CM, engl. Condition Monitoring), tuhoamaton arviointi (NDE, engl. Non-Destructive Evaluation), tilastollinen prosessinohjaus (SPC, engl. Statistical Process Control) ja vaurioennuste (DP, engl. Damage Prognosis). Tyypillisesti SHM yhdistetään rakenteiden, kuten lentokoneiden tai rakennusten, globaaliin etävalvontaan. CM oikeastaan vastaa täysin SHM:ää jo suomenkielisen nimensäkin puolesta, mutta erona on se, että CM keskittyy pyörivien ja edestakaista liikettä tekevien koneistojen valvontaan. Nämä koneistot ovat yleensä jotain teollisuuskoneita tai sähköntuotannossa tarvittavia generaattoreita kuten aiemmin mainittu tuulimyllyn vaihteisto. Tärinämittaus on keskeisessä roolissa tässä toimintatavassa. NDE:n mukaista toimintamallia käytetään vasta paikan päällä, kun vaurio on paikallistettu. Kohteina voi olla esimerkiksi paineastia tai raide. NDE:tä käytetään vaurion vakavuuden kartoitukseen, kun jo tiedetään vian olevan olemassa, ja missä se sijaitsee. Termi tuhoamaton on peräisin arvioinnissa käytettävistä tekniikoista, jotka eivät tuhoa tutkittavaa kohdetta. Tällaiset tekniikat perustuvat esimerkiksi ultraäänen, magneettipartikkeleiden, nestepenetraation, radiografian, elektromagnetismin tai visuaalisten havaintojen käyttämiseen [2]. SPC, kuten nimestäkin voi päätellä, on prosessiperustainen eikä rakenteisiin keskittyvä. Toimintatapa valvoo antureiden avulla muutoksia prosessissa mutta kytkeytyy rakenteiden valvontaan siinä mielessä, että muutos prosessissa johtuu loppuen lopuksi jostain materiaaliviasta tuotantokoneistossa. Viimeisenä, kun vika on havaittu, käytetään DP:tä ennustamaan järjestelmän jäljellä oleva toiminta-aika. Tämä toimintatapa auttaa aikatauluttamaan huoltotoi-

menpiteitä, jos huollon viivytys on ylipäänsä mahdollista. [1, s. 304–305]

2.2 Miksi kunnonvalvontaa tehdään?

Nyt tiedetään, mitä käsite kunnonvalvonta tarkoittaa, joten on aika siirtyä tarkastelemaan, mitä hyötyä kunnonvalvonnalla saavutetaan. Hyötyjä on varmasti moniakin tilanteesta riippuen, mutta käydään tässä läpi kaksi yleisintä. Aloitetaan siitä yleisimmästä eli rahasta. Kuten aiemmin todettiin, kunnonvalvonnan avulla pystytään havaitsemaan vaurioita rakenteissa, ja mahdollisesti ennakoivalla huollolla voidaan jopa välttää lopullinen vikaantuminen, joka estää rakenteen toiminnan. Tästä seuraa suora taloudellinen hyöty sitä kautta, että jos tehtaissa tuotantolaitteisto pystytään pitämään jatkuvasti käynnissä, niin tällöin ei pääse syntymään ikäviä katkoja, jotka tulevat kalliiksi. Suurissa tehtaissa tunninkin katkos voi tuoda jopa miljoonien eurojen menetykset, joten piilevästä vikaantumisesta on ensiarvoisen tärkeää saada tietoa jo etukäteen, jotta varaosat ehditään tilaamaan hyvissä ajoin ennen vikaantumista. Toinen vaihtoehto olisi pitää varalaitteita varastossa vikaantumisien varalta, mutta toisaalta tämäkin lähestymistapa on erittäin kallis, koska tuotantokoneistot eivät yleensä ole halpoja, ja niiden lepuuttaminen käyttämättömänä on turhaa, jos vikaantumisia ei tule ollenkaan tai niitä tulee hyvin harvakseltaan. Kunnonvalvonnan avulla siis tuotanto voidaan optimoida tuottamaan enemmän vähentämällä turhia kustannuksia. [1, s. 305]

Toinen kunnonvalvonnalla saavutettava hyöty on turvallisuus. Ihmisten turvallisuus on aina etusijalla kaikessa tekemisessä. Otetaan esimerkkinä avaruusteknologia, jossa ei haluta ottaa ylimääräisiä riskejä ihmishenkien kustannuksella, mikäli se vain on estettävissä. Avaruussukkuloissa valvotaan oikeastaan jokaista mahdollista asiaa, jotta komentokeskus ja astronautit ovat jatkuvasti tietoisia siitä, mitä tapahtuu. Jos jossain järjestelmässä havaitaan kriittinen vaurio, tehtävä voidaan keskeyttää, ja sukkula tuodaan turvallisesti takaisin maan pinnalle. Jos vaurio ei ole kriittinen, ja se on korjattavissa ennen kuin se pahenee, tehtävää ei tarvitse keskeyttää, vaan vaurio voidaan korjata astronauttien toimesta, minkä jälkeen tehtävä voi jatkua turvallisesti. Toinen esimerkki turvallisuudesta on maanjäristyksen tuhojen vaikutusten tutkiminen rakennuksissa. Kunnonvalvonnan avulla voidaan päätellä, onko ihmisten turvallista palata rakennukseen työskentelemään tai asumaan maanjäristyksen jälkeen. [1, s. 305]

Sekä rahaan että turvallisuuteen liittyvä esimerkki on suuren pääoman vaativat rakenteet kuten lentokoneen rungot, suihkumoottorit ja jättimäisten rakennustyömaiden laitteistot. Tämän kaltaisten rakenteiden valmistajat haluaisivat siirtyä mieluummin vuokrausliiketoimintaan kuin myydä ne kokonaan, koska vuokrausmallilla valmistajat voisivat ottaa vastuun rakenteiden huollosta, jolloin huoltovälit olisivat optimaalisia. Tällöin rakenteet pysyvät turvallisina käyttäjille ja samalla generoivat kentällä rahaa omistajilleen. [1, s. 305–306]

Näistä syistä johtuen kiinnostus kunnonvalvontaa kohtaan on noussut merkittävästi viime aikoina. Yritykset ja julkiset laitokset ovat kiinnostuneita panostamaan turvallisuuteen samalla saaden rahallista hyötyä. Pyörivien rakenteiden kunnonvalvontaan liittyviä konferensseja on pidetty paljonkin, mutta uusiakin konferensseja

on perustettu useampia pelkästään SHM:ään liittyen. [1, s. 306]

2.3 Miten kunnonvalvonta toteutetaan?

Kunnonvalvonta on siis hyödyllistä ja kannattavaa. Seuraavaksi katsotaan, miten kunnonvalvonta on toteutettu ennen, ja mitä vaihtoehtoja nykyään on tarjolla. Tätä kautta päästäänkin tämän työn varsinaiseen aihepiiriin eli M2M-teknologian hyödyntämiseen kunnonvalvonnassa.

Mahdollisesti varhaisinta vauriontunnistamista on tehty akustisiin menetelmiin perustuen. Yksi tällainen menetelmä on esimerkiksi junan pyörään napauttaminen, jolloin pyörän antamasta äänivasteesta voidaan päätellä, onko pyörä kunnossa vai onko siinä jotain vialla. Tämä menetelmä vaatii harjaantuneen korvan ja ihmisen tekemään päätöksen rakenteen kunnosta. Sittemmin ihminen on keksinyt avukseen erilaisia välineitä ja mittalaitteita, joilla vauriontunnistamista on helpotettu. Kehitys on kulkenut käsi kädessä tietokoneiden kehityksen kanssa, sillä mitä enemmän laskentatehoa on saatu yhä pienempään muotoon halvemmalla, se on avannut SHM:lle uusia mahdollisuuksia. [1, s. 306]

Menestyksekkäintä on ollut pyörivien rakenteiden kunnonvalvonta eli CM. CM perustuu siirtymien, nopeuksien ja kiihtyvyyksien mittauksiin, ja niiden historiatietojen vertailuun. Anturit kiinnitetään tiettyyn pisteeseen rakenteessa. Yleensä anturit laitetaan mittaamaan sekä poikittaista että pystysuuntaista poikkeumaa, jotta tietoa saadaan erisuuntaisista tärinöistä. Myös muunlaisia suureita voidaan valvoa kuten lämpötilaa, kierrosnopeutta, painetta, öljyn laatua, käyttötunteja ja liikkeitä. Tehtaissa anturit voidaan kytkeä valvomoon, jossa on jonkinlainen tietokone tai taajuusanalysointiyksikkö tärinäantureita varten. Vauriontunnistus voidaan tässä tapauksessa tehdä niin, että ihminen katselee antureilta saatavaa taajuusspektriä ja vertailee eri aikoina otettuja spektrejä keskenään, minkä perusteella päätös rakenteen kunnosta voidaan tehdä. Toinen tapa on antaa tietokoneen laskea anturin mittaamasta tärinästä nopea Fourier-muunnos (FFT). FFT antaa tuloksena myös taajuusspektrin, josta tietokone voi tehdä päätelmiä joko yksittäisen spektrin perusteella tai vertaamalla sitä tietokantaan kerättyyn historiatietoon. Tyypillisimmät tunnistettavat viat ovat löystyneet tai vaurioituneet laakerit, vinoutunut akseli ja nirhautunut hammaspyörän hammas. [1, s. 306–307]

Yllä mainitulla tavalla saadaan yhden tai mahdollisesti useamman rakenteen valvonta keskitetysti valvomoon, mikäli ne sijaitsevat lähellä valvomoa, mutta mitä jos valvottavat rakenteet sijaitsevatkin kaukana toisistaan ympäri maailmaa ja ovat vielä mahdollisesti liikkuvia tai muuten vain sijoitettu hankalakulkuiseen paikkaan? Tällöin johtojen vetäminen on kannattamatonta, joten jokin langaton teknologia on tarpeen. Aluksi kehitettiin telemetria ratkomaan näitä haasteita. Telemetria perustuu radioaaltojen käyttämiseen viestinnässä. Telemetriaa käytetään julkisesti esimerkiksi avaruussukkuloiden valvontaan. Toinen näkyvästi esillä oleva esimerkki on Formula 1-kilpailut, joissa kilpa-autoista saatavia anturitietoja siirretään langattomasti valvomoon. Näissä kahdessa esimerkissä telemetria toimii hyvin, koska valvottavia kohteita on vähän tai ne ovat paikallisia. Telemetria kuitenkin tarvitsee toimiakseen jokaiselle laitteelle oman taajuuskanavan, jolloin siitä tulee resurs-

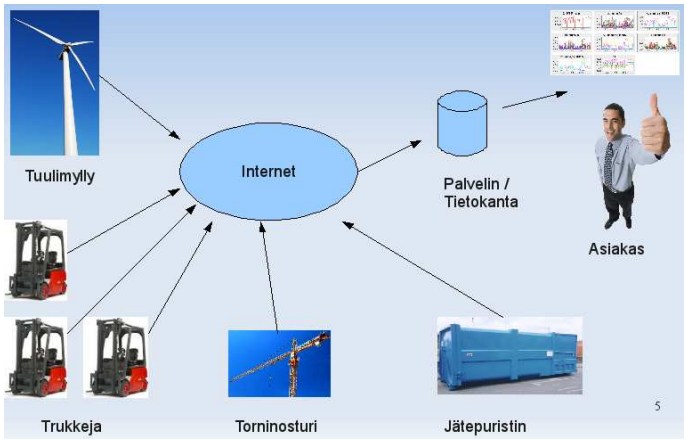
sisyöppö teknologia, kun valvottavia kohteita voi olla ympäri maailma tuhansia tai jopa useita miljoonia, jos otetaan huomioon kaikkien valmistajien valvottavat laitteet. Radiotaajuudet ovat muutenkin niukka julkinen resurssi, joita julkinen hallinto pyrkii jakamaan järkevästi. Tätä ongelmaa silmällä pitäen on kehittynyt käsite koneviestintä eli M2M.

2.4 M2M

Tutustutaan hieman koneviestinnän käsitteeseen eli kuten johdannossa todettiin, termi tulee englanninkielisistä sanoista Machine-to-Machine communication, joka yleisesti lyhennetään M2M:ksi. Se tarkoittaa määritelmän mukaan koneiden välistä viestintää ilman ihmisen vuorovaikutusta. Tyypillisin M2M-alustan tehtävä on toimia tiedonkeruujärjestelmänä. Aloitetaan tutustuminen sillä, että käydään läpi, mitä termi pitää sisällään, ja katsotaan millaista teknologiaa viestinnässä käytetään hyväksi. Lopuksi esitellään M2M:n muita sovellusalueita kuin pelkkä kunnonvalvonta.

M2M ei ole tieteellinen keksintö vaan puhtaasti kaupallinen termi. Tästä syystä aihepiiristä ei ole tieteellisiä kirjoituksia, joten lähdeteoksia on niukalti. Kirjakauppojen verkkosivuilla tehtävät haut aihepiiristä antavat tuloksia vähän tai ei ollenkaan, joten tämän luvun sisältö pohjautuu henkilökohtaiseen tietotaitoon ja kokemukseen, joka on kertynyt lähes neljän vuoden aikana alalla toimimisen johdosta. M2M-teknologia hyödyntää jo olemassa olevia teknologioita ja infrastruktuureita, joten kaikki tieteellinen liittyy alla oleviin teknologioihin. Käydään seuraavaksi läpi näitä teknologioita.

2.4.1 Yleiskuva



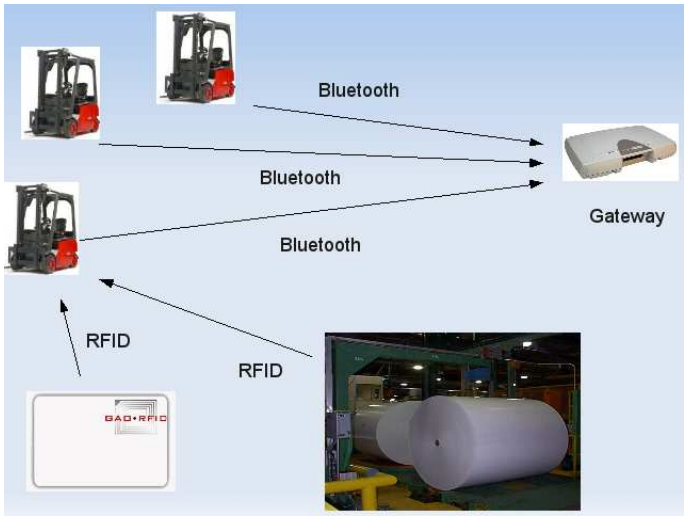
Kuva 1: Yleiskuva M2M-järjestelmästä

Kuvassa 1 on yleiskuva tyypillisestä M2M-järjestelmästä. M2M-järjestelmä koostuu valvottaviin kohteisiin asennettavista pienistä tietokoneista eli päätelaitteista, mahdollisista yhdyskäytävistä (engl. gateway), Internetistä ja palvelimista. Kuvaan

on valittu mukaan kohteita, joita on hankala valvoa normaalein keinoin. Tuulimyllyyn ja torninosturiin on vaivalloista kiivetä, trukit liikkuvat, joten kaapelointi on mahdotonta ja jätepuristimen täyttöaste on vaikea tietää, kun ei näe umpinaisen säiliön perälle. Kaiken tämän voi välttää, jos rakenteisiin asennetaan pieni tietokone, joka lähettää tarvittavaa tietoa langattomasti tai joissakin tapauksissa myös langallisesti Internetin välityksellä palvelimelle, jossa tieto tallentuu tietokantaan. Tietokantaan tallennetut tiedot voidaan esittää verkkosivuilla erilaisina diagrammeina, joista asiakas voi nähdä valvottavien kohteiden tilan. Jos rakenteissa havaitaan vaurio (SHM/CM/SPC) tai muuta epätavallista, tietoa voidaan käyttää hyväksi päätöksenteossa ja jatkotoimenpiteissä. Tietojen perusteella voidaan myös arvioida rakenteen jäljellä oleva toiminta-aika (DP).

M2M-viestintä voidaan jakaa karkeasti kahtia lyhyen ja pitkän kantaman teknologioihin. Lyhyen kantaman teknologioihin kuuluvat Bluetooth, WLAN, RFID ja ZigBee. Pitkän kantaman teknologioihin lukeutuvat GPRS, UMTS, SMS, Ethernet ja GPS. Käydään seuraavaksi tämä lyhenne- ja käsiteviidakko läpi kuvien ja esimerkkien kera.

2.4.2 Lyhyen kantaman teknologiat



Kuva 2: M2M-järjestelmässä käytettäviä lyhyen matkan viestintäteknologioita

Kuvassa 2 nähdään tyypillinen esimerkki lyhyen kantaman viestintäteknologioista. RFID eli radiotaajuinen etätunnistus verhoaa sisäänsä nipun erilaisia standardeja, ja siitä on erilaisia toteutuksia alkaen muutaman sentin toimintasäteestä aina 100 metrin toimintasäteeseen asti [3]. RFID koostuu aktiivisesta lukijasta ja passiivisesta tunnisteesta eli tagista. Kun tagi tuodaan tarpeeksi lähelle lukijaa, tagissa oleva elektroniikka aktivoituu lukijan lähettämän elektromagneettisen säteilyn energiasta, jolloin tagi herää lähettämään heikon signaalin, joka toimii tunnisteena lukijalle. RFID:tä voidaan hyödyntää monella tapaa eri tunnistusetäisyyksien vuoksi, mutta kuvaan on poimittu yksi lyhyen etäisyyden tapaus ja yksi pidemmän etäisyyden

tapaus. Ensimmäisessä tapauksessa kuljettaja voi kirjautua trukkiin näyttämällä omaa kulkukorttiaan trukin lukijaan, jolloin kuljettaja tunnistetaan, ja trukki aktivoituu käytettäväksi. Toisessa tapauksessa tuotteita siirrellessä päätelaite voi lukea RFID:n avulla kyydissä olevan tuotteen tiedot, ja tietojen perusteella voidaan päivittää varastotilannetta reaaliajassa palvelimelle.

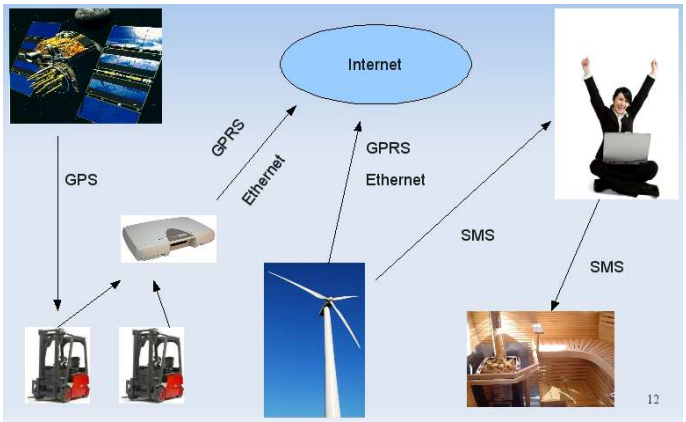
Varastohallissa voi olla myös yhdyskäytävä tai tukiasema, jonka kautta trukit voivat lähettää tietoa palvelimelle. Tällöin tiedonsiirto trukkien ja yhdyskäytävän välillä on mahdollista sarjamuotoisen Bluetoothin tai langattoman lähiverkon eli WLAN:n välityksellä. Yhdyskäytävältä tieto voidaan välittää pitkän kantaman viestintää käyttäen. Yhdyskäytävän käyttäminen ei ole välttämätöntä, koska tieto voidaan siirtää päätelaitteelta palvelimelle suoraan pitkän kantaman teknologioitakin käyttäen. WLAN perustuu IEEE 802.11-standardiin, joka käyttää hyväkseen vapaassa käytössä olevia taajuuksia 2,4 GHz, 4,9 GHz ja 5,0 GHz [4]. WLAN tarjoaa IP-pohjaisen liikennöinnin päätelaitteen ja tukiaseman/yhdyskäytävän välillä. Bluetooth on avoin langaton protokolla, joka toimii WLAN:n tavoin vapaalla 2,4 GHz taajuudella. Bluetooth tarjoaa RS-232-standardin (nimetty myöhemmin EIA-232:ksi) mukaisen sarjaliikennöintiprotokollan mutta langattomasti. Sekä WLAN:n että Bluetoothin tyypillinen kantama on noin 100 metriä riippuen esteistä. Tämä riittää mainiosti yhteyden muodostukseen ja tiedon lähetykseen, kun trukit ajavat yhdyskäytävän ohitse varastohallissa, sillä kokonaisuudessaan trukit voivat olla noin 200 metrin matkan kantavuusalueella.

Uusi tulokas lyhyen kantaman viestinnässä on ZigBee. ZigBee perustuu IEEE 802.15.4-standardiin, joka pyrkii siihen, että ZigBee laitteet käyttäisivät mahdollisimman vähän virtaa viestintään, joten esimerkiksi anturi voisi toimia hyvin pitkiä aikoja (vuosia) samalla paristolla [5]. ZigBeen avulla voidaan muodostaa esimerkiksi paikallinen langaton anturiverkko, jolloin valvottavan rakenteen anturointi ei tarvitse lainkaan kaapelointia, ja lukeminen tapahtuu täysin langattomasti. Tästä on erityisesti hyötyä pyörivän rakenteen anturoinnissa tai suurehkon hallialueen anturoinnissa, koska mitään kaapeleita ei tarvitse asentaa, ja antureita on helppo siirrellä paikasta toiseen tarpeen vaatiessa [6].

2.4.3 Pitkän kantaman teknologiat

Kuvassa 3 nähdään pitkän kantaman teknologioita. Aloitetaan GPS:stä eli satelliittipohjaisesta paikannuksesta. GPS oli alunperin vain Yhdysvaltain armeijan käytössä mutta on sittemmin vapautettu myös siviilikäyttöön. GPS-järjestelmä koostuu kolmesta segmentistä, joita ovat avaruus-, kontrolli- ja käyttäjäsegmentti. Avaruussegmentti käsittää 24 maata kiertävää satelliittia. Kontrollisegmentti pitää sisällään ympäri maailmaa olevia tarkkailuasemia, joilla säädetään ja tarkkaillaan satelliittien toimintaa. Käyttäjäsegmentti koostuu kaikista maailman GPS-vastaanottimista. Satelliittien lähettämän paikannussignaalin avulla vastaanotin voi määrittää tarkan ajan ja kolmiulotteisen paikan maapallolla. Tätä informaatiota voidaan hyödyntää esimerkiksi ajoneuvojen seurannassa. [7]

Kun valvottava tieto on saatu luettua joko satelliitin signaalista tai anturilta, tiedon saamiseksi palvelimelle on valittavissa nippu teknologioita, joista osa on lan-



Kuva 3: M2M-järjestelmässä käytettäviä pitkän kantaman viestintäteknologioita

gattomia, mutta joukossa on myös yksi langallinen teknologia. Yleisin tällä hetkellä käytetty teknologia on perinteisen GSM-verkon päälle toteutettu GPRS, joka tarjoaa pakettikytkentäisen pääsyn Internetiin ja sitä kautta palvelimelle. Samankaltaisen yhteyden Internetiin tarjoaa myös uudempi UMTS-teknologia, joka kansankielellä tunnetaan paremmin nimellä 3G, kun GPRS edustaa nimikettä 2G. G tulee englanninkielisestä sanasta generation, joka tarkoittaa sukupolvea. Sukupolviajattelu on keksitty helpottamaan kansalaisten ymmärtämystä matkapuhelinverkon kehityksestä verhoten alla piilevät teknologiat yleistermin alle. GPRS:n huono puoli on siinä, että puhe ja tiedonsiirto käyttävät verkossa samoja resursseja, ja koska puhe on priorisoitu tiedonsiirtoa korkeammaksi, tiedonsiirtonopeus voi jäädä hyvinkin alhaiseksi, jos tukiaseman alueella on paljon puheluita käynnissä. UMTS:n suunnittelussa on lähdetty siitä ajatuksesta, että puhe ja tiedonsiirto on erotettu toisistaan, jolloin ne eivät häiritse toisiaan. UMTS tarjoaa samalla huomattavasti paremman tiedonsiirtonopeuden verrattuna GPRS:ään. UMTS ei ole kuitenkaan vielä yhtä yleinen tiedonsiirtoteknologia M2M:ssä kuin GPRS. Oliotalo on kokeillut yhden pilottiprojektin yhteydessä tiedonsiirtoväylänä UMTS-verkkoa pääkaupunkiseudulla vuonna 2009, mutta tarjolla olleet USB-liitäntäiset modeemit olivat toiminnaltaan liian epävarmoja, joten tuotantoon siirryttäessä päätettiin kuitenkin palata käyttämään GPRS:ää.

GSM-verkon vanhointa teknologiaa edustaa SMS eli tekstiviestit. Tekstiviestipohjainen tiedonvälitys on M2M:n alkuaikojen teknologiaa mutta edelleen aktiivisesti käytössä joissakin vanhoissa järjestelmissä. Tiedonvälitystapana tekstiviesti on varmatoimisin, koska tekstiviesti saadaan lähetettyä, vaikka signaalivoimakkuus tukiasemaan olisi heikkokin. GPRS ja UMTS tarvitsevat toimiakseen huomattavasti paremman signaalivoimakkuuden kuin tekstiviestin lähetys ja vastaanotto. Matkapuhelinverkko on tänä päivänä kuitenkin levinnyt niin laajalle, että ongelma ei ole enää ajankohtainen. SMS on edelleen kätevää teknologiaa kotiautomaatiossa, koska lähettämällä tekstiviesti esimerkiksi saunan ohjausjärjestelmään, voidaan sauna kytkeä päälle jo hyvissä ajoin ennen kotiin saapumista, mitä kuvan 3 oikeassa laidassa oleva viestintä kuvaa. Tämä esimerkki lipeää hieman ulos koneviestinnän alu-

eelta, koska ihminen osallistuu viestintään, mutta samalla idealla voitaisiin ajastaa taloyhtiön saunavuorot tietokoneohjatusti, jolloin ollaan jälleen koneviestinnän alueella. Rakenteiden valvonnassa tekstiviestejä käytetään tänä päivänä siten, että kun valvontatietoa saapuu palvelimelle, ja mikäli tieto aiheuttaa hälytyksen, palvelimelta voidaan lähettää asiantuntijalle tekstiviesti, jotta hälytykseen voitaisiin reagoida mahdollisimman nopeasti.

Edelliset olivat kaikki langattomia teknologioita, mutta on olemassa myös yksi perinteinen langallinen teknologia, jota käytetään hyväksi, mikäli sellainen on tarjolla valvottavassa kohteessa. Tämä teknologia on Ethernet, joka perustuu IEEE 802.3-standardiin, ja jossa tiedonvälitys tapahtuu kehyksien välityksellä [8] - [12]. Ethernetin avulla voidaan liittyä lähiverkkoon eli LAN:iin ja liikennöidä pakettikytkentäisesti. LAN:sta voidaan yleensä järjestää helposti pääsy Internetiin. Ethernet on järkevä vaihtoehto silloin, kun valvottavassa kohteessa on jo entuudestaan olemassa yhteys LAN:iin ja Internetin, joten tällöin uuden laitteen liittäminen verkkoon ja sen liikennöinti ei maksa juurikaan ylimääräistä. GPRS-liikennöinnissä tiedonsiirron hinta voi määräytyä joillakin sopimustyypeillä tavupohjaisesti, jolloin suuria tietomääriä siirrettäessä hinta voi olla todella kallis. Nykyään GPRS:ään on tarjolla sellaisia sopimuksia, joilla on oikeus liikennöidä rajattomasti kiinteällä kuukausimaksulla. Tämä ei kuitenkaan päde roaming-tilanteessa eli silloin, kun ollaan ulkona kotimaan verkosta, ja vieraillaan ulkomaisen operaattorin verkossa. Roaming-tilanteessa tiedonsiirron hinta määräytyy jälleen tavupohjaisesti, ja hinta on vieläpä kalliimpi kuin kotiverkossa. Ethernet on myös paljon nopeampi kuin langattomat teknologiat, joten Ethernetin välityksellä on mahdollista siirtää suuriakin tietomääriä.

2.4.4 Sovellukset

Nyt tiedetään, millaisia teknologioita M2M verhoaa sisäänsä. Koska tietoa keräävä päätelaite on liitettävissä joustavasti Internetiin, se avaa lukuisia mahdollisuuksia valvoa erilaisia kohteita monella eri teollisuudenalalla. Käydään läpi muutama uusi esimerkki, joissa M2M-alustaa on mahdollista käyttää hyväksi, ja jotka eivät ole esiintyneet vielä tämän työn yhteydessä.

Öljy- ja kaasuteollisuudessa voidaan valvoa virtauksia, paineita, lämpötiloja ja tankkien täyttöasteita [13]. Kiinteistönvalvonnassa voidaan valvoa huoneistojen lämpötiloja ja säätää lämmityslaitteistoja toimimaan optimaalisesti, jotta energiaa säästyisi [14]. Kiinteistönvalvontaan voidaan yhdistää turvallisuustoiminteita kuten kulunvalvontaa ja automaattista ovien lukitusta ja avaamista. Murron tapahtuessa voidaan lähettää koneviestintää hyödyntäen hälytys valvomoon tai jopa suoraan lähimmälle partiolle, jos partioautossa on esimerkiksi paikannusjärjestelmä, jonka perusteella hälytys osataan kohdistaa lähimpään partioon. Myyntiautomaatit sisältävät nykyään enemmän älyä, koska joihinkin automaatteihin voi soittaa saadakseen virvoitusjuoman, ja automaatit voivat kertoa ongelmista tai täyttöasteensa ylläpitäjälle, jotta automaatin tyhjentyessä, tai pikemminkin ennen tyhjentymistä, se voidaan täyttää uudelleen [15]. Terveydenhoidossa voidaan potilaaseen kiinnittää antureita ja lähetinyksikkö, joilla voidaan valvoa elintoimintoja, jolloin raja-arvojen ylittyessä tai alittuessa voidaan hälyttää hoitaja tai lääkäri paikalle [16].

Edelliseen kappaleeseen oli kerätty joitakin sovelluksia toisistaan poikkeavilta aloilta siitä syystä, että se osoittaa, miten joustavasta alustasta on kyse. Palataan hieman edellisen luvun ongelmaan radiotaajuuksien käytöstä. Kuten tämä luku osoittaa, M2M ei ole resurssisyöppö alusta vaan käyttää tehokkaasti hyväkseen jo olemassa olevia teknologioita. Kaikki esitellyt langattomat teknologiat käyttävät omaa tiettyä radiotaajuusaluetta siten, että samanaikainen käyttö monen käyttäjän kesken on mahdollista. M2M:ssä päätelaitteen paikallinen kytkeytyminen Internetiin voidaan tehdä joko kohteessa olevan langallisen teknologian avulla tai sitten tilanteen mukaan yhteys luodaan käyttäen joitakin esiteltyjä langattomia teknologioita. M2M:n suurin vahvuus onkin Internetin maailmanlaajuinen tiedonvälityskyky, jota ei tarvitse luoda aina uudestaan. Toisaalta taas M2M ei ole pelkän Internetin varassa vaan on kykenevä viestimään myös puhtaasti matkapuhelinverkon välityksellä käyttäen tekstiviestejä.

2.5 Yhteenveto

Tämä luku antoi johdatuksen kunnonvalvontaan. Luvussa opittiin, että kunnonvalvonta on vaurion tunnistamista järjestelmästä. Vaurio voi syntyä järjestelmään joko hitaasti tai nopeasti. Luvussa käytiin läpi viisi erilaista vauriontunnistamisstrategiaa, jotka poikkeavat hieman toisistaan. Seuraavaksi lukijalle motivoitiin, miksi kunnonvalvonta on hyödyllistä. Tässä yhteydessä päädyttiin siihen tulokseen, että kunnonvalvonnan suurimpia hyötyjä ovat raha ja turvallisuus. Tämän jälkeen katsottiin, miten kunnonvalvontaa on tehty ennen vanhaan, ja millaisia vaihtoehtoja nykyään on tarjolla. Vanhanaikainen tekniikka perustui rakenteiden nakutteluun, jolloin äänivasteesta saatiin viitteitä rakenteiden kunnosta. Kehitys on tuonut tarjolle antureita ja pieniä tietokoneita, joilla rakenteiden kuntoa voidaan valvoa.

Lopuksi esiteltiin M2M-teknologia, joka edustaa kehityksen huippua kunnonvalvonnassa. Luvussa avattiin M2M-käsitettä sen verran, että saatiin ymmärrys siitä, mitä kaikkea termi verhoaa sisäänsä, ja miten se auttaa kunnonvalvonnan haasteisiin. M2M jakautui sekä lyhyen että pitkän kantaman teknologioihin. Tarjolla olevat teknologiat käytiin läpi, minkä jälkeen esiteltiin M2M-teknologian muita sovellusalueita. Tässä vaiheessa huomattiin, että M2M-teknologia soveltuu hyvin joustavasti erilaisiin tehtäviin.

3 M2M-alustan nykyinen rakenne

Nyt on avattu M2M käsitettä sen verran, että voidaan sukeltaa syvemmälle katsomaan, millaiseen ratkaisuun Oliotalo on käytännössä päätynyt oman M2M-alustan kanssa. Tässä luvussa esitellään Oliotalon M2M-alustan kaikki tämän työn kannalta oleelliset komponentit, jotta voidaan jatkossa ymmärtää, millaisia tekijöitä järjestelmässä on, mihin kohtaan järjestelmää haasteet kohdistuvat ja mihin kohtaan järjestelmää tämän työn ratkaisu tullaan toteuttamaan. Oliotalon M2M-alusta noudattaa edellisessä luvussa esiteltyä rakennetta, eli valvottavassa kohteessa on asennettuna päätelaite, joka eri teknologioita käyttäen lähettää valvontatietoa palvelimelle tietyn ennalta sovitun protokollan mukaisesti. Avataan jokainen komponentti yksitellen auki, ja tarkastellaan niiden rakennetta lähemmin.

3.1 Yleistä

Otetaan alkuun lyhyt yleiskatsaus Oliotalon tuoteperheeseen, ja siirrytään sitten yksityiskohtaisempaan esittelyyn. Oliotalon M2M-alustan nimi on CuteAnimals (söpöt eläimet), joka juontaa nimensä siitä, että jokaisella alustan komponentilla on jokin eläimen nimi. Päätelaite on saanut nimen Platypus (vesinokkaeläin). Oliotalo on kehittänyt oman ohjelmointikielen nimeltä Hedgehog (siili), joka on Lisp-johdannainen kieli. Hedgehog-ohjelmisto sisältää sekä kääntäjän että tulkin. Hedgehog-ohjelmisto on avointa lähdekoodia, ja sen on suunnitellut aikoinaan Lars Wirzenius Oliotalolla työskennellessään, mutta nykyisen toteutuksen on luonut Kenneth Oksanen [17]. Korjauksia ja lisäyksiä Hedgehog-ohjelmistoon ovat täydentäneet Markus Andersson ja tämän työn kirjoittaja. Päätelaitteessa ajettava mittausohjelmisto on kirjoitettu Hedgehog-kielellä. Mittausohjelmiston nimeäminen poikkeaa eläinteemasta, sillä sen nimi on dOGMA, jonka isot kirjaimet tulevat sanoista Oliotalo General Measurement Application, eli kyse on yleisestä mittaussovelluksesta, joka tarjoaa rungon muillekin tuotteille. dOGMA-ohjelmistoa käytetään pääasiassa kunnonvalvontaan (CM) mutta sisältää paljon perusteknologiaa viestintään liittyen. dOGMA:sta johdannaiset ohjelmistotuotteet ovat kuitenkin eläinten nimiä, ja ne kulkevat nimellä Rhino (sarvikuono), Orca (miekkavalas), Raccoon (pesukarhu) ja Giraffe (kirahvi). Rhino on yleinen työkoneiden valvontaohjelmisto, jossa painopiste on käyttöaikojen valvonnassa. Orca on erikoistunut trukkien valvontaan, jolloin lisänä Rhinoon tulevat kuljettajantunnistus ja törmäykset. Raccoon on kehitetty jätepuristimien täyttöasteen valvontaan. Giraffe lähettää valvontatietoa torninosturin liikkeistä ja välittää virhekoodeja torninosturin tietokoneelta. Palvelimesta löytyy ohjelmisto nimeltä Beaver (majava), jonka tehtävänä on toimia viestintärajapintana päätelaitteiden ja palvelimen sovellusohjelmiston välissä. Yhdyskäytäväohjelmiston nimeksi on päätynyt Squirrel (orava). Oliotalon käyttämä protokollakin on saanut eläinaiheisen nimen, koska se on tärkeä osa M2M-tiedonvälitystä. Protokollasta puhutaan nimellä Sparrow (varpunen).

Kaikkiin näihin ei perehdytä tämän syvällisemmin, mutta nostetaan esille työn kannalta muutamia tarpeellisia komponentteja, joita lähdetään tarkastelemaan hieman tarkemmin. Poimitaan tarkasteluun Platypus eli itse päätelaite, siihen liittyvä

ohjelmisto dOGMA, koska se sisältää viestintään liittyvät toiminnot, Beaver palvelinpuolelta, koska se on palvelimen varsinainen komponentti, joka kommunikoi päätelaitteen kanssa ja viimeisenä Sparrow, koska se on keskeinen viestinnän välikappale.

3.2 Päätelaitteen fyysinen olemus

Platypus on Oliotalon kehittämä pieni tietokone, jolla voidaan suorittaa etävalvontatehtäviä. Se on suunniteltu siten, että sillä pystyy lukemaan yleisimpiä antureiden antamia ulostulosignaaleja ja väyläarkkitehtuureja. Lisäksi Platypuksella on mahdollista kontrolloida ulkoisia piirejä olemaan auki tai kiinni. Tätä ominaisuutta voidaan käyttää hyväksi tilanteissa, joissa valvottavassa laitteessa havaitaan jokin kriittinen ongelma, jolloin laite tai jokin osa siitä saadaan sammumaan nopeasti, ettei suurempaa vauriota ehdi syntymään. Tärkeimpänä ominaisuutena M2M:n kannalta Platypuksessa on monipuoliset tietoliikennekytkennät.

Taulukko 1: Platypuksen tekniset tiedot.

Käyttöjännite	12 – 48 VDC
Suoritin	ARM-9 198 Mhz
Keskusmuisti	64 Mt
Massamuisti	NAND-flash 256 Mt
A/D-muunnin	12-bittinen, 16 kanavaa
Viestintä / Tietoliikenne	USB RS-232 RS-485 CAN iButton GSM (GPRS) UMTS (USB:n kautta) Ethernet (USB:n kautta) Bluetooth RFID (RS-232:n tai USB:n kautta) WLAN (USB:n kautta) GPS
Ulostulot	Audio-ulostulo, VGA-ulostulo, 3 kpl open drain-tyyppisiä ulostuloja, 2 kpl digitaalisia ulostuloja

Taulukosta 1 nähdään Platypuksen teknisiä tietoja. Käyttöjännitteeksi on valittu alue, joka on yleensä saatavissa valvottavassa kohteessa. 12 – 48 VDC on suoraan saatavissa ajoneuvojen akuista, koska tyypillisesti henkilöautoissa akun jännite on

12 VDC, kuorma-autoissa 24 VDC ja trukeissa 24 tai 48 VDC. Suoritin perustuu ARM:n suunnittelemaan 32-bittiseen suoritinarkkitehtuuriin, jonka tehonkulutus on minimaalista verrattuna tavallisiin pöytätietokoneiden suorittimiin, koska nykyiset pöytätietokoneiden suorittimet kuluttavat tehoa jopa yli 100 W, kun Platypuksen koko elektroniikka kuluttaa vain alle 4 W tehoa perustuen Oliotalon omiin mittauksiin [18]. Tämän ominaisuuden ansiosta Platypus soveltuu hyvin akkukäyttöiseksi, koska se voi olla toiminnassa myös silloin, kun ajoneuvo ei ole käynnissä, eikä se syö akkua tyhjäksi sillä aikaa, kun ajoneuvo on sammuksissa.

Laitteessa on 64 Mt keskusmuistia, joka riittää mainiosti Linux-käyttöjärjestelmän ja sovellusohjelmistojen suorittamiseen. Laitteessa on vakiona 256 Mt massamuistia, jonne tallennetaan kriittistä mittaustietoa, jonka halutaan säilyvän muistissa vaikka sattuisi virtakatkos. Myös sovellusohjelmisto on tallennettu massamuistille.

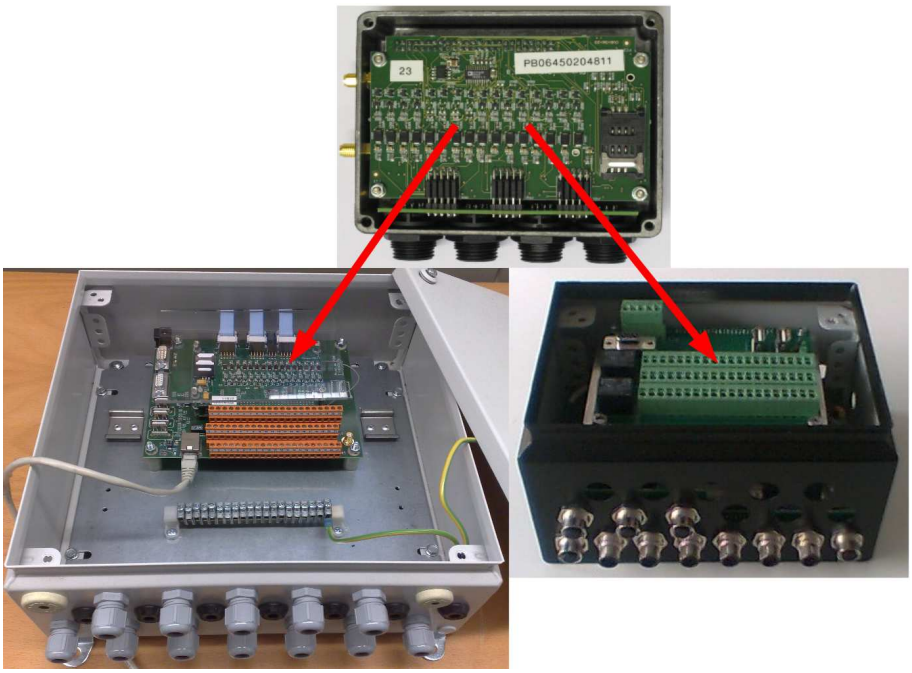
Analogista mittaustietoa voidaan lukea analogi/digitaali-muuntimen (lyhyemmin A/D-muunnin) avulla. Siinä on 16 kanavaa, ja jokaisen kanavan jännitetieto saadaan 12-bittisenä eli eri jännitetasoja on täten 4096. Kanavat voidaan konfiguroida joko alueelle 0 – 5 VDC tai 0 – 10 VDC. USB on kätevä yleisväylä, jonka avulla voidaan liittää monia oheislaitteita. Näin Platypuksen laitteistoa voidaan kasvattaa helposti ilman, että tarvitsee suunnitella uutta elektroniikkaa. RS-232- ja RS-485-väylien avulla voidaan lukea erilaisia antureita ja laitteita. Esimerkiksi RS-232-väylään voidaan liittää RFID-lukija ja RS-485-väylän välityksellä voidaan lukea Modbus-anturilenkkiä. Modbus-lenkkiin voidaan liittää 246 anturia, joissa jokaisessa voi olla maksimissaan 65536 eri mitattavaa suuretta, joten laajennusmahdollisuus on merkittävä [19, s. 7] ja [20, s. 7]. CAN-väylä on yleinen tiedonsiirtoväylä ajoneuvokäytössä. CAN-väylän välityksellä on mahdollista ohjata ajoneuvon laitteita ja valvoa niiden toimintaa [21, s. 1 – 22]. iButton on teknologia, jolla luetaan RFID:n tapaan tunnisteita. Tunnisteen lukemisen perusteella voidaan tehdä jokin haluttu toiminne.

Viestintävaihtoehdot ovat pitkälti samat kuin edellisen luvun esittelyssä. ZigBee on ainoa viestintäteknologia, joka puuttuu. Osa vaihtoehdoista löytyy suoraan piirikortilta kuten GSM-modeemi, Bluetooth ja GPS, mutta osa on mahdollista liittää esimerkiksi USB:n välityksellä.

Platypuksesta löytyy audio-ulostulo, joka tarkoittaa sitä, että kytkemällä ulkoisen kaiuttimen, voidaan Platypukselta soittaa mitä tahansa ääntä. Tätä ominaisuutta käytetäänkin tällä hetkellä vanhuksille suunnatussa ruoka-automaatissa, joka ohjeistaa ja informoi käyttäjäänsä puhumalla. VGA-ulostulon avulla Platypuksesta saadaan kuvasignaali ulkoiselle näytölle. Tätä ominaisuutta voidaan hyödyntää asennuskoestimen kytkentään. Asennuskoestin voi olla esimerkiksi kosketusnäyttö, jolla asentaja voi tarkistaa, että anturikytkennät on oikein tehty, ja yhteys palvelimeen on kunnossa. Open drain-tyyppisillä ulostuloilla voidaan ohjata releitä, joilla edelleen ohjataan ulkoisia sähköpiirejä. Digitaaliset ulostulot on lähinnä tarkoitettu LED:ien ohjaamiseen, mutta sovelluksia löytynee niin paljon kuin mielikuvitusta riittää, sillä digitaalinen ulostulo on teknisesti niinkin yksinkertainen kuin 5 VDC syöttö päällä tai pois. Samoin myös open drain-ulostulo mahdollistaa monia sovelluksia.

Platypuksen toimintalämpötila on alueella -40 – +100 °C ellei GSM-modeemia

kalusteta mukaan. Mikäli GSM-modeemi on mukana, toimintalämpötila kaventuu alueelle -20 – +70 °C, koska GSM-modeemeja ei yksinkertaisesti valmisteta laajemmalle lämpötila-alueelle. Vaikka modeemi kalustetaan mukaan, ympäristö saa siitä huolimatta olla hyvin vaihteleva, joten Platypus kykenee toimimaan erittäin haastavissa olosuhteissa.



Kuva 4: Platypus ja räätälöidyt asiakaskohtaiset ratkaisut

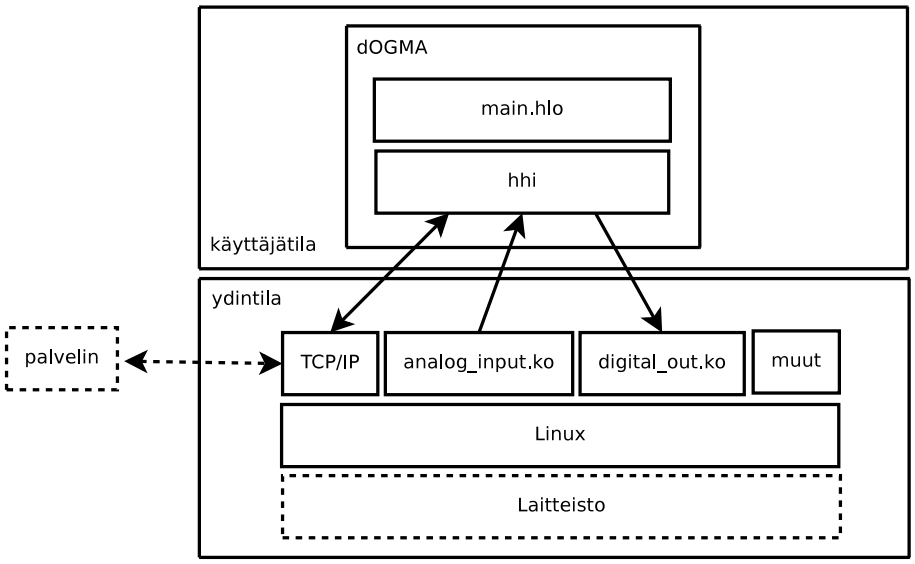
Kuvassa 4 ylimpänä nähdään Platypuksen peruskokoonpano. Elektroniikka koostuu kahdesta levystä, jotka on pinottu toistensa päälle. Tämä pino on koteloitu alumiiniseen koteloon, josta kytkennät tuodaan läpi liittimien avulla. Sekä kotelo että liittimet ovat IP68 suojattuja, mutta antennin läpivienti pudottaa luokituksen IP65:ksi. IP-luokitus (engl. International Protection) määrittelee sähkölaitteiden tiiveyden. Merkintä koostuu kirjaimista IP ja kahdesta numerosta, joista ensimmäinen kertoo suojan pölyä vastaan ja toinen vesitiiveyden. Tässä tapauksessa numero 6 tarkoittaa täydellistä suojaa pölyä vastaan, ja numero 5 tarkoittaa vesisuihkun kestävyyttä. Numero 8 tarkoittaa täydellistä vesitiiveyttä, eli tuotteen voi upottaa yli metrin syvään veteen pysyvästi, eikä vettä saa päästä kotelon sisälle [22]. Koska tämän työn yhteydessä puhutaan myös verkkoteknologiasta, IP-lyhennettä ei tule sekoittaa Internet Protocol lyhenteen kanssa. Korkean IP-luokituksen ansiosta tällainen Platypuksen kotelointi sopii hyvin likasiin ja kosteisiin ympäristöihin kuten ajoneuvoihin.

Platypuksen peruselektroniikasta on jalostettu myös muita tuotteita siten, että on suunniteltu yksi tai useampi lisälevy, johon on laitettu USB-keskitin, Ethernet-piiri, releitä, VGA-liitin ja riviliitin kytkennöille. Kuvan 4 kahdessa alemmassa versiossa osa analogi-kanavista on varustettu siten, että niillä voidaan lukea virtasig-

naalia normaalin jännitesignaalin asemesta. Tämä on saatu aikaan lisäämällä vastus maapotentiaalin ja luettavan virtasignaalin väliin, jolloin A/D-muuntimella voidaan lukea vastuksen yli oleva jännite, josta edelleen sähköopin kaavojen avulla voidaan laskea virran suuruus. Antureiden antamat virtaviestit ovat normaalisti 0 – 20 mA alueella. Alempien versioiden osa kanavista on kalustettu myös lukemaan Pt-100 lämpötila-antureita. Pt-100 anturin platina reagoi lämpötilan muutoksiin siten, että sen resistanssi kasvaa lämpötilan kasvaessa ja päinvastoin, joten lämpötilan selvittämiseksi on ensin mitattava resistanssin suuruus, minkä jälkeen resistanssi muutetaan joko taulukon tai muunnoskaavan avulla lämpötilaksi. Resistanssin lukeminen ei onnistu suoraan jännitemittauksella, vaan sitä varten täytyy olla Wheatstonesilta, jonka avulla jännite-eroista voidaan laskea resistanssin suuruus. Jos siis halutaan lukea jotain muuta suuretta kuin jännitettä, lisälevylle on lisättävä riviliittimen ja Platypuksen A/D-muuntimen väliin erinäisiä komponentteja, jotta eri suureiden mittaus voidaan palauttaa loppuen lopuksi jännitemittaukseksi. Näitä jalostettuja versioita Platypuksesta käytetään tuulimyllyjen vaihteistojen valvontaan.

3.3 Päätelaiteohjelmiston rakenne

Nyt laitteistoon on saatu hieman tartuntapintaa, joten voidaan siirtyä tarkastelemaan laitteistolla suoritettavia ohjelmistokerroksia. Ohjelmiston voi karkeasti jakaa kahtia, jolloin toinen osa koostuu käyttöjärjestelmästä, ja toinen osa muovautuu sovellusohjelmistosta. Käyttöjärjestelmässä ei sinänsä ole mitään mielenkiintoista, joten luku keskittyy enemmänkin sovellusohjelmiston läpikäymiseen.



Kuva 5: Päätelaitteen ohjelmistokerrokset

Kuvasta 5 nähdään, että lähimpänä laitteistoa on käyttöjärjestelmä, joka tässä tapauksessa on Linux tai joissakin yhteyksissä käytetään myös merkintää GNU/Linux, jolloin halutaan ilmaista, että se sisältää GNU-projektin kehittämää vapaata

ohjelmistoa. Tällä hetkellä Linux-ytimestä on käytössä versio 2.6.29, mutta kentällä on myös 2.6.22 versiollisia ytimiä. Tässä päästäänkin jouhevasti tämän työn ongelmaan käsiksi, sillä 2.9.22 versio sisältää kriittisen ongelman, joka saa aikaan tiedostojärjestelmän lukittumisen, jolloin laite lakkaa toimimasta. Ongelma liittyy JFFS2 tiedostojärjestelmään ja tapaukseen, jossa mihin tahansa tiedostoon paloittainen tiedon lisääminen saa aikaan sellaisen tilanteen, että tiedostojärjestelmä lukittuu käyttökelvottomaksi [23]. Tämä ongelma on korjattu Linux-ytimen versioon 2.6.29. Tästä seuraakin se, että kentällä oleviin päätelaitteisiin on tarkoitus päivittää uudempi ydin, jotta lukittumista ei tapahtuisi. Nykyään Oliotalo on siirtynyt käyttämään uusissa toimituksissa paremmaksi havaittua UBIFS tiedostojärjestelmää, mutta vanhoihin kentällä oleviin päätelaitteisiin ei voida tehdä etänä tiedostojärjestelmän muutosta, koska vanha tiedostojärjestelmä pitäisi pyyhkiä pois, jolloin tietoa häviää, ja toisaalta puhdas asennus täytyy tehdä ulkoiselta medialta kuten USB-muistilta, jota ei voida siirtää paikan päälle tietoverkkoa pitkin, vaan operaatio vaatisi asentajan käynnin paikan päällä.

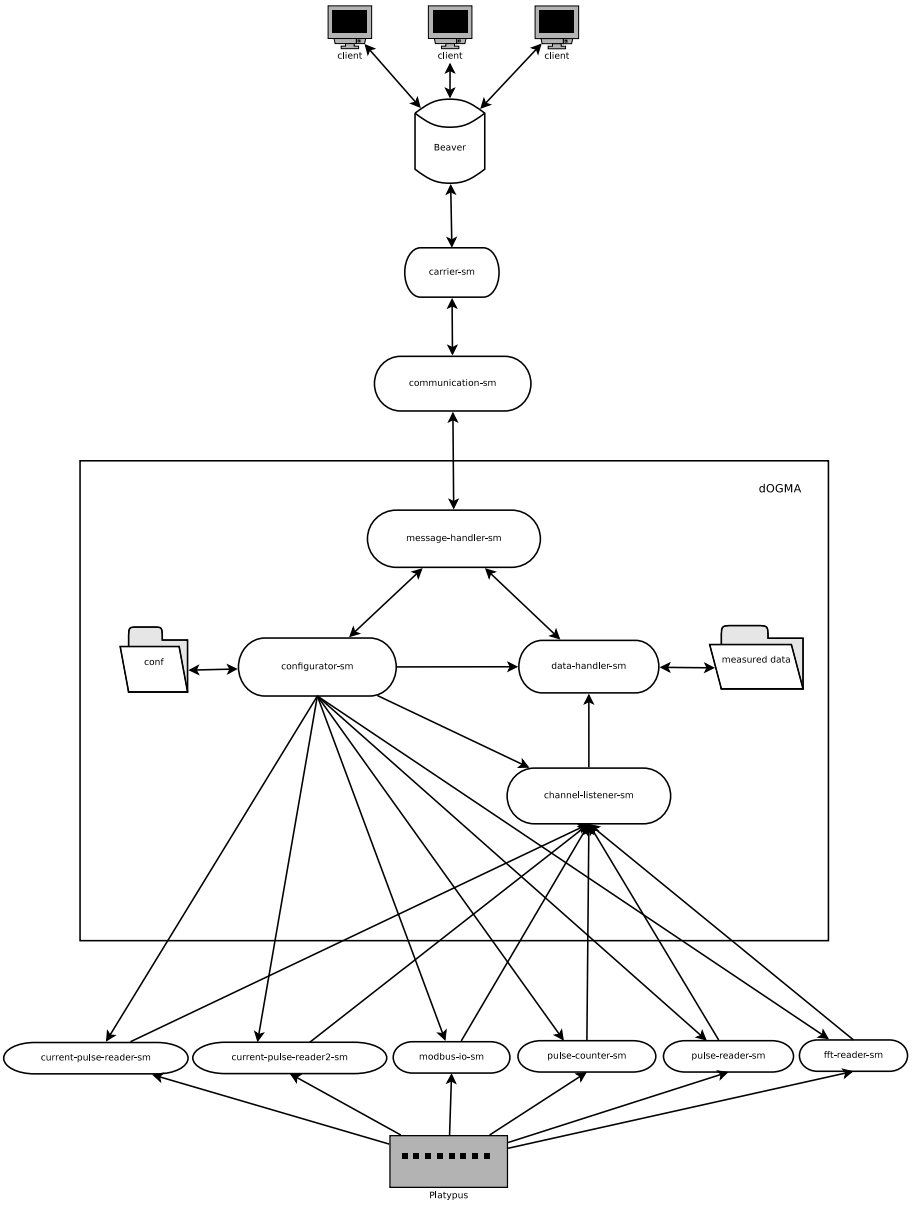
Vaikka nykyisistä Linux-ytimistä löytyykin valmiiksi laiteajuriohjelmat suurelle joukolle laitteita, Platypus sisältää kuitenkin muutamia komponentteja, jotka eivät ole kovin yleisiä. Tällaisia komponentteja ovat esimerkiksi A/D-muunnin ja digitaalisia ulostuloja ohjaava piiri. Näitä vastaavat ajurit ovat kuvassa järjestyksessä analog_input.ko ja digital_out.ko. Paljon muitakin ajureita tarvitaan, mutta jätetään ne tässä yhteydessä luetteloimatta yksinkertaisuuden vuoksi. Edellä mainitut ajurit toimivat niin sanotussa ydintilassa, jossa on mahdollisuus käskyttää laitteita suoraan ruohonjuuritasolla. Tämä on kuitenkin monimutkaista ja vaatii laitteiston tuntemusta, minkä takia ohjelmointivirheitäkin on helppo tehdä kuten aiemmin mainittu JFFS2:n lukkiutumisvirhe. Tästä syystä käyttöjärjestelmä tarjoaa käyttäjille turvallisemman käyttäjätilan, jossa ohjelmille annettavat muistialueet ovat suojatumpia kuin ydintilassa, ja ajurit tarjoavat yksinkertaisemman rajapinnan laitteiston käskyttämiseen. Käyttäjätilassa on mahdollista suorittaa normaaleja sovellusohjelmistoja kuten tämän työn tapauksessa mittaus- ja viestintäohjelmistoa.

Seuraavaksi kuvassa on käyttäjätilassa suoritettava Hedgehog-tulkki hhi, jolla suoritetaan mittausohjelmiston tavukooditiedostoa main.hlo. Tämä mittausohjelmisto on aiemmin mainittu dOGMA. dOGMA käyttää analog_input.ko ajurin tarjoamaa rajapintaa lukeakseen analogi-antureilta luvuiksi muutettuja jännitetasoja. Samoin dOGMA käyttää hyväkseen digital_out.ko ajurin tarjoamaa rajapintaa digitaalisten ulostulojen ohjaamiseen. Kun mittaustieto on saatu luettua analogiajurilta dOGMA:n muistiin, tieto täytyy siirtää palvelimelle, ja tämä onnistuu käyttöjärjestelmän tarjoaman TCP/IP rajapinnan avulla. TCP:llä voidaan luoda luotettava tiedonsiirtoputki päätelaitteen ja palvelimen välille. Kun putki on avattu, sen välityksellä voi lähettää ja vastaanottaa tavuja. Tämän putken välityksellä liikennöidään Oliotalon Sparrow-protokollan mukaisesti, mutta TCP:stä kerrotaan lisää myöhemmin.

Otetaan seuraavaksi käsittelyyn syvällisemmin itse dOGMA. Hedgehog-kielen suunnittelun lähtökohtana on ollut tehdä mahdollisimman helpoksi ohjelmoida tilakoneabstraktioita. Tilakoneiden avulla on helppo mallintaa erilaisia automaatteja. Tilakoneajattelu on helppo mieltää lyhyen esimerkin avulla. Kuvitellaan, että pitäisi

mallintaa virvoitusjuoma-automaatin toiminta tilakoneen avulla. Aluksi pitää olla jossain alkutilassa eli automaatti odottaa kolikon pudotusta (tai nykyään myös soittoa). Kun kolikko on pudotettu, automaatti siirtyy odottamaan käyttäjältä virvoitusjuoman valintaa. Kun käyttäjä painaa haluamansa virvoitusjuoman valintanappulaa, automaatti siirtyy tilaan, jossa se käynnistää moottorin pudottaakseen virvoitusjuoman jakeluluukkuun käyttäjän saataville. Tämän jälkeen automaatti siirtyy jälleen alkutilaan odottamaan uutta kolikon pudotusta. Tarvittavien tilojen määrä vaihtelee luonnollisesti ongelman mukaan, mutta automaatin tulee aina olla deterministinen äärellinen automaatti eli tilojen määrä samassa automaatissa on vakio, ja tilasiirtymät on tapahduttava aina ennaltamäärätyn säännön mukaan. dOGMA rakentuu monesta tällaisesta tilakoneesta. Hedgehog-kielessä tilakoneiden välillä on mahdollista lähetellä viestejä, joiden vaikutuksesta voidaan suorittaa toiminteita.

Kuvassa 6 nähdään dOGMA:n ja siihen liitännäisten tilakoneiden suhdeverkosto. Kuvassa alimpana nähdään tilakoneita, jotka lukevat A/D-muuntimelta tai eri väyliltä raakatietoa, jota mahdollisesti jalostetaan esimerkiksi tunnistaen erimuotoisia pulsseja tai yksinkertaisesti laskevat pulsseja, minkä jälkeen tieto lähetetään prosessoituna ylöspäin channel-listener-sm-tilakoneelle. Channel-listener-sm-tilakoneessa luetaan itsessään normaaleja jännite-, virta- ja Pt-100-kanavia. Channel-listener-sm-tilakoneessa tehdään kaikki älykäs päättely hälytyksistä, digitaalisten ulostulojen ohjaamisesta ja tiedon tallentamisesta. Mikäli tieto päätetään tallentaa, hieman ylempänä data-handler-sm vastaa kaikenlaisesta mittaustiedon säilömisestä, ja kun on tiedon lähetyksen aika, tallennettu tieto haetaan lähetettäväksi. Tieto lähetetään message-handler-sm-tilakoneelle, joka muodostaa protokollan mukaisen viestin ja lähettää sen edelleen communication-sm-tilakoneelle. Communication-sm on eräänlainen viestipuskuri, joka säilöö lähetettäviä viestejä, mikäli palvelimeen ei ole yhteyttä. Jos palvelimeen on yhteys, viestipuskurin viestejä yritetään lähettää palvelimelle. Communication-sm lisää lähetyksen yhteydessä aikaleiman jokaiseen viestiin, jotta palvelimella viestit voidaan sijoitella oikeaan järjestykseen. Sen jälkeen viesti koodataan tavujonoksi, joka lähetetään carrier-sm-tilakoneelle, jonka kautta viesti lähetetään Oliotalon Beaver-palvelimelle. Carrier-sm-tilakone pyrkii aktiivisesti olemaan jossain verkkoyhteydessä. Yleisimmin yhteystapana on GPRS tai Ethernet. Jos toinen yhteystapa katkeaa, yritetään vaihtoehtoisesti toista. Tätä jatketaan, kunnes yhteys syntyy jommalla kummalla yhteystavalla. Kun yhteys Internetiin on auki, carrier-sm pyrkii muodostamaan TCP-yhteyden palvelimeen. Päätelaite on siis aktiivinen osapuoli viestinnässä. Koska TCP tarjoaa kaksisuuntaisen yhteyden, myös palvelimelta käsin voi lähetellä viestejä kuten konfiguraatiota tai pyytää niin sanotusti on-demand mittaus eli tilata jokin erillinen mittaus pyynnöstä. Tällöin viestit kulkeutuvat luonnollisesti päinvastoin, eli carrier-sm ottaa tavujonoja vastaan ja lähettää niitä communication-sm:lle, joka purkaa tavujonot viesteiksi, jonka jälkeen matka voi jatkua message-handler-sm:lle, jossa viestin sisältö tutkitaan, ja viesti ohjataan oikealle tilakoneelle. Jos viesti on konfiguraatio, se lähetetään configuration-sm:lle, joka tallentaa konfiguraation massamuistille, ja samalla se puretaan auki, minkä jälkeen jokaiselle tilakoneelle jaetaan tarvittavat asetukset. Vaikka kaikki tilakoneet ovatkin samassa ohjelmassa, vain älykäs osa ohjelmistoa muodostaa dOGMA:n, koska muut tilakoneet voitaisiin tarpeen vaatiessa



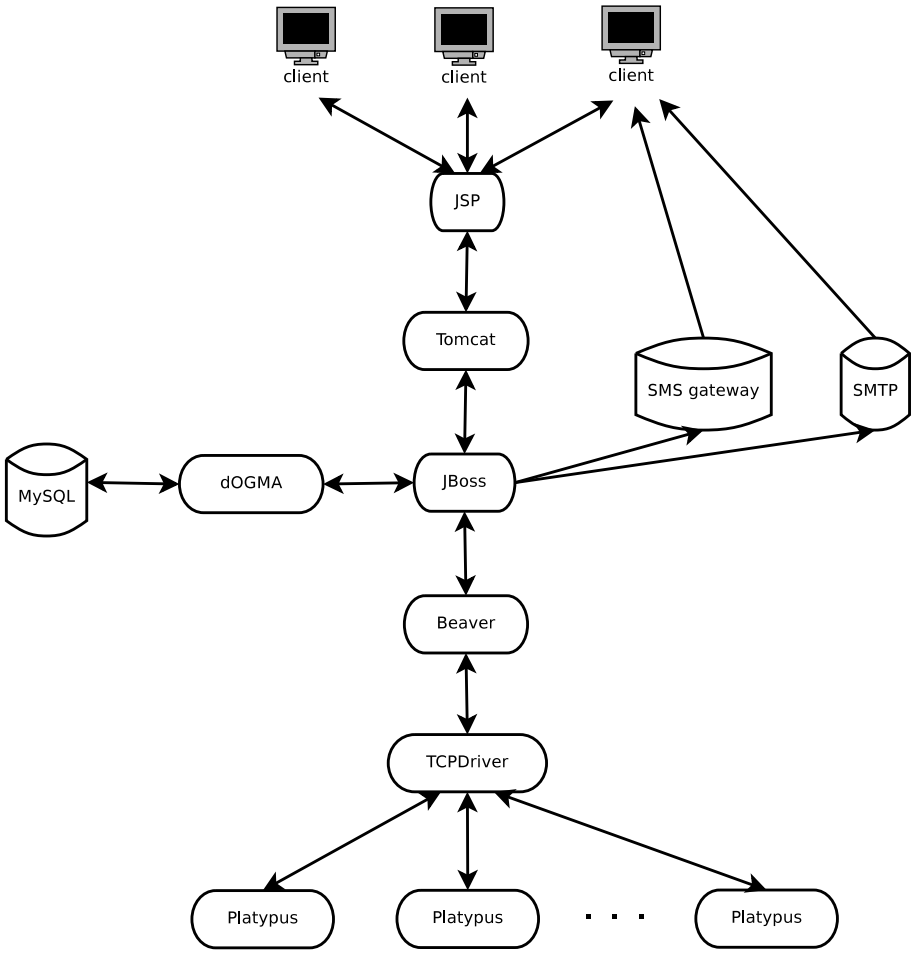
Kuva 6: dOGMA:n tilakoneabstraktio

erottaa omiksi ohjelmikseen, ja viestinvälitys tilakoneiden välillä voitaisiin hoitaa tiedostorajapintojen kautta.

3.4 Palvelinohjelmiston rakenne

Palvelinohjelmisto koostuu JBoss-ohjelmiston ympärille toteutetuista palveluista. JBoss on Red Hat, Inc. yhtiön tekemä vapaa ja modulaarinen väliohjelmisto, joka tarjoaa viestinvälitysarkkitehtuurin eri palveluiden välille. JBoss ei sellaisenaan vielä tee mitään järkevää, koska tarkoituksena on luoda niin sanottu liiketoimintaäly (engl. business intelligence) JBossin ympärille. Tässäkin tapauksessa Oliotalo

on luonut JBossin ympärille oman liiketoimintaälynsä, joka käsittää mittaustiedon tallentamisen ja esittämisen asiakkaille. JBoss on Java-kielellä ohjelmoitu ohjelmistopaketti, joten luonnollisesti myös palvelimen palvelut on ohjelmoitu Java-kielellä. JBoss toteuttaa EJB 3.0 määritelmän, jonka ansiosta kaikki käsiteltävä tieto voidaan mallintaa Java-kielen olioina. Tämä nopeuttaa kehitystyötä, koska raaka kanssakäyminen tietokannan kanssa vähentyy siitä syystä, että ohjelmistokehittäjä voi luoda Java-mallisen olion ja käskyttää väliohjelmistolle, että tallenna olio tietokantaan, jolloin väliohjelmisto järjestää tiedon järkevässä muodossa tietokantaan. Samoin tietoa haettaessa voidaan käskyttää väliohjelmisto hakemaan tietty olio tietokannasta, jolloin väliohjelmisto hakee tiedot tietokannasta ja muodostaa uudelleen tallennetun olion. Väliohjelmiston ansiosta tietokanta pysyy vakaassa tilassa, koska tiedon tallennus ja haku tehdään hyvin testattujen rutiinien kautta. Käydään palvelinohjelmiston rakenne palveluittain läpi kuvan avulla.



Kuva 7: Palvelinohjelmiston rakenne

Kuvassa 7 nähdään palvelinohjelmiston rakenne. Edetään kuvassa alhaalta ylös. Kun päätelaitteet ottavat yhteyttä palvelimeen, ensimmäisenä tervehtii TCPDriver, jonka tehtäviin kuuluu hyväksyä päätelaitteen yhteys, minkä jälkeen liikennöin-

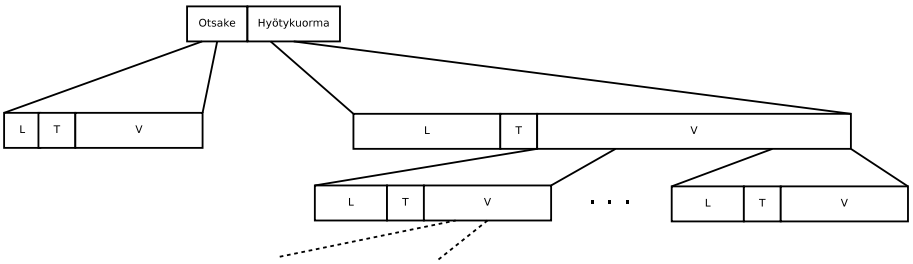
ti päätelaitteen kanssa on mahdollista. Ketjun seuraava on Beaver, joka on hieman vastaava viestipuskuri kuin päätelaitteen communication-sm. Beaver hallinnoi liikennettä päätelaitteiden ja dOGMA-sovelluksen välillä. Jos päätelaitteelle yritetään lähettää esimerkiksi konfiguraatiopäivitys juuri silloin, kun päätelaite ei ole yhteydessä palvelimeen, viesti säilötään Beaverin päätelaitekohtaiseen jonoon. Kun päätelaite ottaa jälleen yhteyden palvelimeen, viestit lähetetään jonosta päätelaitteelle. Kun päätelaite (Platypus) lähettää valvontatietoa, Beaver katsoo otsakkeesta sovellustunnisteen ja välittää viestin sen perusteella oikealle sovellukselle. Tässä tapauksessa oikea sovellus on dOGMA. Viesti kulkeutuu Beaverilta dOGMA:lle, joka tallentaa tiedon tietokantaan. Tietokantana käytetään vapaata MySQL-tietokantaa, joka sijaitsee fyysisesti erillisellä palvelinkoneella, mutta se voisi sijaita yhtä hyvin myös samalla palvelinkoneella dOGMA:n kanssa. Mikäli valvontatieto aiheuttaa hälytyksen, ja asiakas on asettanut itselleen tekstiviesti- ja sähköposti-ilmoitukset, lähetetään asiakkaalle tekstiviesti SMS gateway-palvelun kautta tai vastaavasti sähköposti SMTP-palvelua käyttäen.

Asiakkaan on mahdollista käydä tarkastelemassa valvontatietoa milloin tahansa kirjautumalla verkkosivustolle, jossa valvontatieto tarjoillaan erilaisten kuvaajien avulla. Tätä varten tarvitaan WWW-palvelin, joka tarjoilee verkkosivuja asiakkaan selaimelle HTTP-protokollaa käyttäen. JBoss tarjoaa Tomcat-palvelun, joka toteuttaa WWW-palvelimen toiminnallisuuden. Tomcat toteuttaa Java Servlet ja Java-Server Pages (JSP) määritelmät. JSP:n avulla verkkosivut voidaan luoda dynaamisesti. Staattiset eli muuttumattomat verkkosivut kuvataan HTML-kielellä. Kun tällainen sivu haetaan selaimella, sivu on jokaisella hakukerralla samanlainen. JSP:n avulla HTML-komentojen sekaan voidaan lisätä ohjelmarivejä, jotka muuttavatkin sivun sisältöä annettujen määreiden mukaisesti. Otetaan esimerkiksi käyttäjä, joka haluaa katsella ensin laitteen A valvontatietoja. Käyttäjä valitsee valikosta laitteen A ja käskee hakea tämän tiedot näkyviin. Tämän käskyn saatuaan palvelin hakee dOGMA:n kautta tietokannasta laitteen A tiedot. Nyt verkkosivu muodostetaan dynaamisesti siten, että osa tiedosta on staattista, mutta osa sivusta täytetäänkin tietokannasta saaduilla tiedoilla. Seuraavaksi käyttäjä haluaakin nähdä laitteen B valvontatiedot, jolloin palvelin hakee jälleen tiedot tietokannasta, minkä jälkeen sivu muodostetaan samaan tapaan, mutta tällä kertaa osa tiedoista onkin täytetty laitteen B tiedoilla eikä laitteen A tiedoilla. Verkkosivun ohjelmarivit ovat kuitenkin täysin samat, mutta JSP-määreiden avulla osa tiedosta on vaihtuvaa eli dynaamista ja osa staattista.

3.5 Protokollan rakenne

Oliotalon käyttämä Sparrow-protokolla on TLV-muotoinen. TLV tulee sanoista Type-length-value. Type tarkoittaa tyyppiä, joka ilmoittaa tiedon tyypin. Length eli pituus ilmoittaa, kuinka monella tavulla tieto on ilmaistu. Value eli arvo on varsinainen tieto, joka on siis pituudeltaan edellisen kentän ilmoittama määrä tavuja. Sparrow-protokolla on kuitenkin käännetty niin päin, että ensin tulee pituus dynaamisesti koodattuna yhdellä tai useammalla tavulla, minkä jälkeen tulee tyyppi yhdellä tavulla ja viimeisenä varsinainen tieto yhdellä tai useammalla tavulla. Dy-

naaminen koodaus tarkoittaa sitä, että pituuskenttä voi olla itse pituudeltaan yhden tai useamman tavun riippuen tietokentän pituudesta. Dynaaminen koodaus antaa vapauksia tiedon pituudelle, koska tiedonsiirtotarve vaihtelee yhdestä tavusta moneen miljoonaan tavuun. Näin ei tarvitse arvuutella, kuinka paljon tavuja tietokentän pituuden ilmaisemiseen aina ja ikuisesti tullaan tarvitsemaan. Toinen merkittävä hyöty dynaamisesta koodauksesta on se, että täten vältetään turhaa yleiskulua (engl. overhead) lyhyillä viesteillä, koska jos jokaisen tietokentän pituus ilmaistaan varmuuden vuoksi esimerkiksi neljällä tavulla, ja lyhyiden tietokenttien pituudet voitaisiin ilmaista ainoastaan yhtä tavua käyttäen, jokaisen elementin yhteydessä lähetetään kolme tavua turhaa tietoa, joka kumuloituu luonnollisesti pitkällä aikavälillä. Varsinkin GPRS-yhteyttä käytettäessä ei ole järkevää lähettää turhia tavuja, koska lähetyskapasiteetti on pieni, ja varsinkin roaming-tilanteessa turhien tavujen lähetys vaikuttaa suoraan hintaan, koska laskutus on tavupohjaista. Avataan protokollan rakenne kuva avulla, jotta hahmottaminen olisi helpompaa.



Kuva 8: Sparrow-protokollan rakenne

Kuvassa 8 nähdään Sparrow-protokollan rakenne. Lähetettävä tieto muodostuu siis edellämainituista elementeistä, joiden muoto on TLV tai oikeammin LTV. L- ja V-kentät on tarkoituksella kuvattu eri mittaisina, jotta niiden dynaamisuus tulisi samassa kuvassa esille. Tieto on jaettu tietoliikenteessä tyypilliseen otsake – hyötykuorma jaotteluun. Otsakkeeseen laitetaan kohdistustietoja kuten päätelaitteen sarjanumero, aikaleima, sovelluksen tunniste ja mahdollinen kuittauspyyntö. Jos viesti sisältää kuittauspyynnön, vastaanottavan osapuolen on lähetettävä kuittausviesti, kun tieto on turvallisesti tallessa, jolloin kuittauksen saapuessa viestin lähettänyt osapuoli voi poistaa viestin, koska tietää sen olevan tallessa. Jos kuittausta ei saavu määritellyn ajan puitteissa, viesti yritetään lähettää uudestaan. Hyötykuorma sisältää varsinaista valvonta- tai konfigurointitietoa. Protokolla on myös hierarkkinen, koska LTV-elementtejä on mahdollista olla sisäkkäin, eli kuten kuvasta nähdään, V-kenttä voi jälleen sisältää useita LTV-elementtejä. Tällaisella rakenteella voidaan kuvata hyvin monipuolista tietoa.

3.6 Yhteenveto

Tähän päättyy ketju, joka alkoi päätelaitteella tapahtuvista sähköisistä muutoksista. Tietoa jalostetaan ja analysoidaan reitin varrella siten, että valvottavasta kohteesta selviää juuri oikeita ominaisuuksia, ja loppuen lopuksi tieto on kulkenut Oliotalon

M2M-alustan läpi aina asiakkaalle asti tarkasteltavaksi. Eri kuvissa on tarkoituksella ollut mukana aina hieman ketjun seuraavaa komponenttia, jotta lukijalle hahmottuisi paremmin, kuinka ketju jatkuu jouhevasti eteenpäin. Vaikka tämä vaikuttaa jo toimivalta järjestelmältä, asiakkaat haluavat jatkuvasti uusia ominaisuuksia ohjelmistoon, jolloin syntyy päivitystarve jo kentällä oleviin päätelaitteisiin. Tämäkään ei ole ongelma, koska Oliotalon M2M-alustassa on ohjelmiston etäpäivitysominaisuus, mutta tätä kautta voidaan päivittää vain käyttäjätilan ohjelmistoa ja ajurimoduuleja. Tämä ei riitä, jos virheellisesti toimiva ohjelmakoodi sijaitsee itse käyttöjärjestelmässä eli Linux-ytimessä kuten aiemmin esille nostettu tiedostojärjestelmän lukittusmisongelma. Tällaisia ongelmia varten Oliotalo haluaa tutkia mahdollisuutta luoda huoltoyhteys päätelaitteille keskitetysti. Huoltoyhteyden muodostaminen ei kuitenkaan ole niin suoraviivaista, koska päätelaitteet ovat aktiivinen osapuoli ja päättävät, milloin ovat yhteydessä palvelimeen, ja toisaalta Internetissä on teknologioita, jotka häiritsevät vapaata viestintää koneiden välillä.

4 Verkon häiritsevät elementit

Aloitetaan tämä luku esittelemällä, miten TCP-yhteys luodaan verkossa, jossa ei ole mitään esteitä viestiliikenteelle, ja tarkastellaan yleisesti, miten liikennöinti Internetissä tapahtuu verkkotasolla. Sivutaan samalla OSI-mallia, jotta saadaan kokonaiskuva siitä, millä tasolla asioista puhutaan milläkin hetkellä. Lopuksi siirrytään tarkastelemaan, millä keinoin liikennöintiä voidaan estää tarkoituksenmukaisesti. Tämä luku auttaa ymmärtämään, miksi päätelaite on aktiivinen osapuoli, ja millaisia rajoituksia huoltoyhteyden toteutuksessa tulee ottaa huomioon.

4.1 Liikennöinti Internetissä

TCP-yhteys luodaan kolmivaiheisella kättelyllä. Palvelimella luodaan ensin pistoke (engl. socket), ja liitetään se haluttuun porttinumeroon, minkä jälkeen palvelin kuuntelee passiivisesti tulevia yhteyksiä. Myös asiakas aloittaa luomalla pistokkeen, minkä jälkeen pistoketta käsketään yhdistymään palvelimen IP-osoitteeseen ja haluttuun porttinumeroon. Kättely alkaa sillä, että asiakas lähettää palvelimelle SYN-paketin, johon palvelin vastaa SYN-ACK-paketilla. Viimeiseksi asiakas lähettää palvelimelle vielä ACK-paketin, minkä jälkeen yhteys on luotu, ja kaksisuuntainen viestintä voi alkaa [24, s. 29 – 36]. Kun pistokkeeseen kirjoitetaan tavuja, ne kääritään kullekkin verkolle sopivan kokoisiin IP-paketteihin, jotka maailmanlaajuinen reititysverkosto Internet välittää perille. IP-paketit ovat kuin postipaketteja, koska niiden otsakkeesta löytyy vastaanottajan ja lähettäjän IP-osoite, ja Internet on kuin postilaitos, joka välittää näiden tietojen perusteella paketteja lähettäjältä vastaanottajalle. Internet koostuu reitittimistä, jotka suorittavat pakettien välitystä. Reitittimet ovat tietoisia omasta sijainnistaan verkon hierarkiassa tiettyyn pisteeseen asti. Ne tietävät aina vähintäänkin sen, minne suuntaan IP-paketti tulee seuraavaksi välittää. Saatuaan paketin reititin tarkistaa otsakkeesta saajan IP-osoitteen ja lähettää sen verkossa eteenpäin lähemmäksi vastaanottajaa. Kun reitin varrella jokainen reititin tekee samanlaisen operaation, paketti päätyy lopulta vastaanottajalle. IP-paketit eivät välttämättä saavu järjestyksessä, joten vastaanottajan on järjesteltävä paketit ensin oikeaan järjestykseen, jonka jälkeen paketit puretaan, ja lähetetyt tavut kootaan jälleen yhtenäiseksi tavujonoksi. Tämä tavujono on luettavissa vastaanottajan pistokkeesta täysin samanlaisena kuin matkaan lähtiessä. TCP-yhteys voidaan sulkea monella eri tavalla. Sulkeminen voi tapahtua nelivaiheisesti siten, että molemmat itsenäisesti lähettävät FIN-paketin, johon molemmat vastaavat toisilleen ACK-paketilla, minkä jälkeen yhteys on suljettu. Lisäksi yhteys voidaan sulkea kolmivaiheisesti siten, että kone A lähettää FIN-paketin koneelle B, johon kone B vastaa FIN-ACK-paketilla, ja lopuksi kone A lähettää vielä ACK-paketin, jolloin molemmat sulkevat yhteyden. Linuxin TCP-toteutus lähettää pelkän RST-paketin, jonka jälkeen yhteys katkeaa välittömästi. [25, s. 86 – 87]

IP-paketit voivat kuitenkin joskus hävitä reitin varrella. Normaalisti häviämisen syynä on verkon ruuhka eli tilanne, jossa liikennettä on enemmän kuin reitittimet ehtivät välittää. Reitittimissä on pakettipuskuri, jonne paketteja taltioidaan siksi aikaa, kunnes ne ehditään käsitellä. Pakettipuskuri auttaa hetkellisten purskeiden

käsittelyssä mutta ei kestä tilannetta, jossa paketteja virtaa jatkuvasti nopeammin kuin niitä ehditään käsittelemään. Tällöin puskuri tulee täyteen, ja saapuva paketti joudutaan yksinkertaisesti pudottamaan pois liikenteen seasta. TCP:ssä on ruuhkanhallintamekanismi tällaisten tilanteiden varalle, joten se tarjoaa huolettoman ja luotettavan viestintäväylän. TCP toimii siten, että jokaiseen lähetettyyn pakettiin vastapuolen täytyy lähettää kuittaus ACK-paketilla, jotta lähettäjä voi varmistua sen perillemenosta. Jokaisella lähetettävällä paketilla on oma järjestysnumero, ja kun paketeille vielä lasketaan odotettavissa oleva kuittauksen paluuaika estimoidun kiertoajan perusteella, järjestysnumeroiden epäjatkuvuudesta voidaan päätellä, että paketti on hävinnyt. Mikäli paketin häviäminen havaitaan, se lähetetään uudestaan. Pakettien häviäminen on siinä mielessä normaalia, koska TCP:n liikennöintitapa perustuu siihen, että mikäli paketteja menee varmasti perille, lähetettävien pakettien määrää lisätään, jolloin jossain vaiheessa verkon kapasiteetti tulee vastaan, ja paketteja häviää. Tämä on merkki TCP:lle, että ollaan kapasiteetin rajoilla, jolloin lähetettävien pakettien määrää vähennetään hetkellisesti. Ruuhkanhallintaan on muutama erilainen algoritmi, mutta niihin ei tässä työssä syvennytä sen enempää.

Kuva 9: OSI-mallin kerrokset

Internet koostuu seitsemästä eri kerroksesta. Kerroksittainen malli on nimeltään OSI-malli. Yllä mainittu TCP-yhteys kuuluu kerrokseen neljä, ja porrasta alempana on IP. OSI-mallin kaikki kerrokset näkyvät kuvasta 9. Alimpana on fyysinen kerros, jossa tieto kulkee sähköisinä signaaleina. Seuraavaksi on siirtokerros, jonka tehtävänä on jaotella tietoa hieman älykkäämmin kuin alemmalla kerroksella mutta kykenee siirtämään tietoa vain paikallisessa verkossa. Verkkokerroksella tieto osataan jo siirtää lähiverkosta muihin verkkoihin. Kuljetuskerros tarjoaa jo luotettavuutta tiedonsiirtoon, kuten jo aiemmin nähtiin TCP:n esittelyn yhteydessä, että paketteja osataan järjestellä oikeaan järjestykseen ja lähetellä uudestaan, jos niitä katoaa. Istunto-, esitystapa- ja sovelluskerros ovat jo hieman häilyviä, mutta yleisesti voidaan todeta, että käyttäjien käyttämät ohjelmat toimivat juuri näillä kolmella ker-

roksella. Kerrosmaisen ajattelun ideana on se, että kullakin kerroksella keskitytään hoitamaan oma vastuualue eikä olla oikeastaan edes kiinnostuneita, miten alemman kerroksen teknologia toimii, vaan pääasia on, että se vain toimii, kun sitä käytetään.

Aiemmin mainittiin, että paketteja voi kadota ruuhkan toimesta, mutta pakettien hävittäminen voi olla myös täysin tietoista ja tahallista. Esimerkiksi monet organisaatiot haluavat suojata oman lähiverkkonsa luvattomalta käytöltä siten, että sinne tai sieltä pois päin sallitaan vain tietynlainen liikennöinti. Tällöin organisaation lähiverkon ja Internetin väliin sijoitetaan palomuuri, joka tarkkailee liikennettä ja päättää sille asetettujen säännöstöjen mukaan, minkälainen liikenne sallitaan ja mihin suuntaan. Esimerkkinä voidaan ottaa tapaus, jossa lähiverkon koneilla sallitaan verkkosivujen hakeminen Internetistä, mutta Internetistä käsin ei voi ottaa yhteyttä lähiverkossa olevaan koneeseen. Organisaatiolla saattaa olla myös omat verkkosivut, joita isännöi erillinen kone lähiverkossa. Tälle koneelle on tällöin tehtävä palomuuriin sääntö, että pelkästään verkkosivupalvelimelle on mahdollista ottaa yhteys Internetistä. Näin muut lähiverkon koneet ovat suojassa, koska niihin ei yksinkertaisesti voi ottaa yhteyttä. Asia ei kuitenkaan ole näin mustavalkoinen kuten uutisia seuraamalla on voinut todeta, että tietomurtoja tehdään paljon Internetissä, vaikka murretuissa kohteissa olisi palomuureja ja muita hyväksi todettuja turvamekanismeja. Katsotaan seuraavaksi tarkemmin, millaisia liikenteenrajoitusteknologioita on olemassa.

4.2 Pakettisuodatuksesta sovellustasolle

Ensimmäisen sukupolven palomuurit perustuivat pakettisuodatukseen. Pakettisuodatus tapahtuu OSI-mallin kolmella ensimmäisellä kerroksella. Suodatus tapahtuu siten, että IP-paketeista tutkitaan otsaketietoja kuten portti, lähde- ja kohdeosoite tai eri lipukkeita, ja verrataan näitä tietoja asetettuihin sääntöihin. Jos vastaavuus löytyy, pudotetaan paketti. Pakettisuodatuksen etu on sen nopeus, koska se perustuu pelkkien IP-pakettien otsakkeiden tutkimiseen, joten tehdäkseen suodatuspäätöksiä palomuurin ei tarvitse nousta kerroksissa ylöspäin tutkimaan niiden kuuluvuutta johonkin tiettyyn liikennevirtaan, eikä hyötykuorman sisältöön puututa lainkaan. Tämä on toisaalta myös puute, koska liikennevirtoja ei pystytä erittelemään sovellustai käyttäjäkohtaisesti. Jos samaa konetta, jolla on siis yksi IP-osoite, käyttää useampi käyttäjä, ja johonkin palveluun halutaan rajata pääsy käyttäjäkohtaisesti, niin tämä järjestely ei onnistu pakettisuodatuksella, koska pääsy voidaan joko sallia tai evätä koneen IP-osoitteen perusteella. Pakettisuodatuksen haittapuoleksi voidaan myös lukea suodatussääntöjen hankala ja virheherkkä kirjoittaminen palomuurille. Pahimmaksi puutteeksi voidaan laskea pakettisuodatuksen helppo kierrettävyys. IP-osoite on helppo väärentää, koska mikään ei vaadi lähettäjää laittamaan paketin otsakkeeseen omaa IP-osoitettaan, vaan kenttä voidaan täyttää jollain palomuurissa sallitulla IP-osoitteella, jolloin palomuuri päästää paketin läpi kyseenalaistamatta sen oikeellisuutta. Samaan kategoriaan liittyy välityspalvelimien eli proxyjen käyttö. Tästä on oma luku alempana, mutta todetaan tässä vaiheessa, että välityspalvelimen kautta voidaan hakea tietoa siten, että asiakas pyytää välityspalvelimelta jotain tietoa, jonka palvelin hakee edelleen jostain tietystä paikasta, ja saadessaan vastauk-

sen palvelin välittää sen tietoa kysyneelle asiakkaalle. Jos tietoa pyydetään joltain koneelta, joka on palomuurin takana, pyyntö näyttää palomuurin näkökulmasta siltä, että se tulee välityspalvelimelta eikä sen takana olevalta oikealta kysyjältä. Kun välityspalvelimen kautta hakee useampi kone tietoa, kaikki pyynnöt näkyvät tulevan yhdestä ja samasta IP-osoitteesta. Tätä voidaan väärinkäyttää esimerkiksi siten, että jos johonkin palveluun on määritelty maakohtainen rajoitus IP-osoiteavaruuden perusteella, joku voi pystyttää siihen maahan välityspalvelimen, jonka kautta muista maista päästään hyödyntämään rajoitettua palvelua vapaasti, koska pyynnöt näyttävät tulevan välityspalvelimelta, jonka IP-osoite on sallittujen listalla. [26, s. 13 – 14]

Yllä on kuvattu tilatonta pakettisuodatusta. Myöhemmin on kehitetty tilallinen pakettisuodatus parantamaan tietoturvaa. Tilallinen tarkoittaa sitä, että IP-paketeista tutkitaan myös hyötykuorman sisältö eli esimerkiksi TCP-liikenne. Palomuuri voidaan asettaa sallimaan liikenne tietystä osoitteesta ainoastaan silloin, kun TCP-yhteys on avattu oikeaoppisesti SYN-paketilla. Tällöin yhteys on tilassa yhdistetty. Ilman tilallista analyysia on mahdollista lähettää ACK-vastauksia palomuurin taakse eli siis väärennöksiä kuittauksista, mutta tilallinen analyysi poistaa tämän mahdollisuuden, koska jos palomuurin takaa ei ole edes yritetty avata yhteyttä, niin palomuurin ulkoa ei silloin pitäisi tulla kuittauksiakaan. ACK-vastauksilla hyökkääjän on mahdollista kartoittaa palomuurin takana olevia koneita ja niissä toimivia palveluita. [26, s. 15]

Myös sovellustasolle asti on kehitetty palomuureja. Näiden etuna on, että ne valvovat kaikkia OSI-mallin kerroksia, ja täten voidaan asettaa hyvinkin yksityiskohtaisia rajoituksia liikenteelle. Sovelluspalomuurin ansiosta voidaan esimerkiksi sallia HTTP-liikenne ja kieltää FTP-liikenne. Nykyään paljon yleistyneet henkilökohtaiset palomuurit, jotka valvovat yksittäisessä koneessa, mahdollistavat vielä senkin, että tietyt ohjelmat voidaan estää liikennöimästä. Sovelluspalomuurien huono puoli on siinä, että kun uusi protokolla keksitään, täytyy odottaa, että palomuuria kehittävä organisaatio julkaisee uuden version. Toinen huono puoli on tehokkuus. Sovelluspalomuurit eivät ole tehokkaita, koska ne joutuvat tutkimaan ja vertailemaan sääntöjä jokaisella OSI-kerroksella. [26, s. 8 – 11]

Nämä pakettisuodatuksen asettamat rajoitteet ovat yksi syy, minkä takia Oliotalon M2M-alusta on suunniteltu siten, että päätelaitteet ottavat yhteyttä palvelimeen eikä toisinpäin. Kuten yllä todettiin, yhteyden muodostaminen ulkoa palomuurin läpi suojattuun lähiverkkoon on mahdotonta laillisin keinoin. Asiakkaiden lähiverkot ovat joskus hyvinkin tiukasti konfiguroituja, jolloin myös liikennöinti lähiverkon sisältä ulos on kielletty. Tällöin ainoaksi vaihtoehdoksi jää pyytää verkon ylläpitäjää tekemään salliva sääntö tietylle päätelaitteen IP-osoitteelle ja portille.

4.3 Välityspalvelin

Välityspalvelin (engl. proxy server) välittää nimensä mukaisesti tietoa. Välityspalvelin toimii normaalisti kahden koneen välissä välittäen pyydettyä tietoa. Edellisessä luvussa hieman jo sivuttiinkin välityspalvelimen ideaa, joka on sellainen, että asiakas pyytää tietoa välityspalvelimelta, jolloin välityspalvelin hakee tiedon jostain palve-

lusta ja saadessaan vastauksen välittää sen asiakkaalle. Pyyntö voi olla esimerkiksi verkkosivu. Välityspalvelimen tulee toteuttaa sekä asiakkaan että palvelimen toiminnallisuus. Välityspalvelin tarjoaa monia etuja kuten vähentää verkkoliikennettä, lisää turvallisuutta tai mahdollistaa käytönvalvonnan ja -eston. [27]

Web proxy on keskittynyt hakemaan ja taltioimaan verkkosivuja. Tämän tyyppinen välityspalvelin ei ainoastaan hae sivuja vaan myös tallentaa niitä välimuistiin. Näin voidaan vähentää verkkoliikennettä, koska jos jokin toinen kone haluaa hakea samaa tietoa kuin joku muu on hakenut jo aiemmin, voidaan verkkosivu tarjoilla suoraan web proxyn välimuistista, eikä sivua siis tarvitse hakea uudestaan varsinaisesta palvelusta. Näin menetellessä on kuitenkin huonokin puoli, koska välimuistista tarjoillut sivut voivat olla vanhentuneita, jos sivujen tieto uusiutuu tiheään tahtiin. Kun web proxyyn lisätään sisällönsuodatus, voidaan organisaatiossa kieltää tietyn tyyppisten verkkosivujen haku. Verkkosivuja voidaan suodattaa URL:n, DNS:n, MIME:n ja avainsanojen perusteella [28]. URL- ja DNS-suodatus tarkoittaa sitä, että välityspalvelimelle voidaan muodostaa niin sanottu musta lista sivuja ja osoitteita, joita ei suostuta välittämään. Jos mustalle listalle määritellään osoite www.esimerkki.fi, ja tältä koneelta pyydetään hakemaan sivu, pyyntöä ei suoriteta vaan näytetään pyytäjälle viesti, jossa ilmoitetaan, että sivun hakeminen on kielletty. Samalla periaatteella toimii MIME-suodatus. MIME-suodatus perustuu tietyn tyyppisen sisällön kieltämiseen. Mustalle listalle voidaan määritellä esimerkiksi videotiedostot, jolloin niiden lataaminen ei onnistu. Avainsana-suodatus kurkistaa verkkosivujen sanalliseen sisältöön ja löytäessään tekstistä vastaavuuden mustan listan vähintään yhteen sanaan, evätään sivun haku jälleen. Sisällönsuodatusta käytetään yrityksissä ja kouluissa. Yrityksissä halutaan estää työntekijöiden pääsy mahdollisesti haitallisille sivustoille tai muuten vain kohteisiin, jotka eivät ole työtehtävien kannalta tarpeellisia kuten viihdesivustot. Kouluissa halutaan myös estää haitalliset sivustot ja lisäksi alaikäisille sopimattomat sivustot. Välityspalvelin voi olla jopa asennettu siten, että jokaisella verkon käyttäjällä on henkilökohtaiset tunnukset, joilla pitää ensin kirjautua välityspalvelimelle, jotta pääsy Internetiin aukeaa. Näin voidaan valvoa käyttäjäkohtaisesti, millaisilla sivuilla kukin vierailee, tai voidaan mahdollisesti käyttäjäkohtaisesti rajoittaa pääsy tiettyihin osoitteisiin. Esimerkkitilanne voisi olla sellainen, että johtajalla on pääsy laajempaan osoiteavaruuteen kuin alaisilla. Täyden valvonnan välityspalvelimien vastakohdaksi on kehitetty anonyymipalvelimia, jotka peittävät täysin käyttäjien jäljet. Anonyymipalvelin ei jätä muistiin mitään käyttäjien hakemia sivuja eikä pidä kirjaa kyselijöiden IP-osoitteita. Anonyymipalvelimien kanssa tulee kuitenkin olla tarkkana, koska tällaisen palvelimen on voinut pystyttää vihamielinen taho, joka on tarjoavinaan anonymiteettiä mutta tallentaakin todellisuudessa kaiken liikenteen muistiin, josta voidaan urkkia käyttäjien tunnuksia ja salasanoja eri palveluihin. Yleisesti ottaen kirjautuminen välityspalvelimen kautta sivustolle, joka vaatii käyttäjätunnuksen ja salasanan, tulee välttää mahdollisuuksien mukaan. Aina ei ole edes mahdollista tietää, onko verkossa välityspalvelin vai ei, koska se voi olla täysin läpinäkyvä käyttäjälle. Välityspalvelimen olemassaolon voi havaita pelkästään siten, että kaikkien osoitteiden tai protokollien käyttö ei ole mahdollista, vaikka mitään näkyvää estettä toiminnalle ei olisikaan.

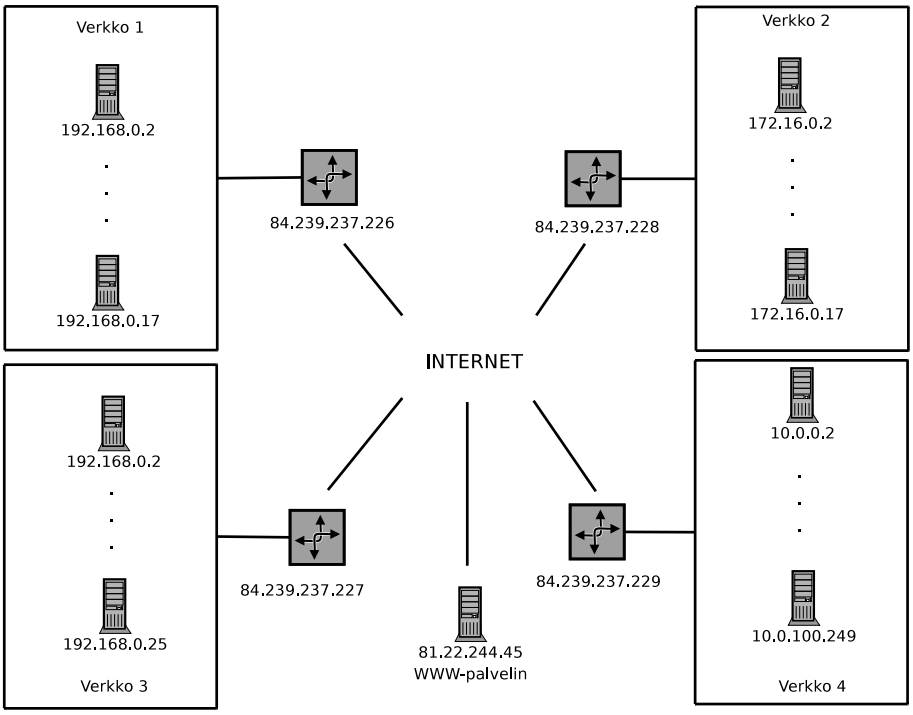
Välityspalvelimet estävät tehokkaasti M2M-viestiliikenteen, koska ne eivät välitä

kuin tietynlaista liikennettä. Web proxyn tapauksessa liikenne täytyy olla HTTP-protokollan mukaista, eli liikenne tulee koostua selkeistä sivujen noutopyynnöistä. TCP-yhteys pystytään luomaan näennäisesti oikein päätelaitteen ja palvelimen välille, mutta yhteys syntyy todellisuudessa vain päätelaitteen ja välityspalvelimen välille. Kun päätelaite yrittää liikennöidä Sparrow-protokollan mukaisesti palvelimelle, liikenne kilpistyykin välityspalvelimelle, josta liikenne ei välity oikealle palvelimelle asti. Tähän ongelmaan Oliotalo on törmännyt monessa tuulipuistossa, jossa asiakas haluaisi suosia Ethernet-yhteyttä kulujen minimoimiseksi, mutta se ei onnistu, joten yhteystapa joudutaan usein vaihtamaan GPRS-yhteydeksi. Roaming-tilanteessa tämä tarkoittaa mittavia kustannuksia. Monesti suomalainen SIM-kortti vaihdetaan paikalliseen korttiin kulujen minimoimiseksi.

4.4 Osoitteenmuunnos

IPv4 on nykyisin laajimmalti käytössä oleva Internetin osoitteistus, ja se on käymässä ahtaaksi, sillä sen osoitteet ilmaistaan vain 32-bitillä, jolloin teoreettisia osoitteita on 2^{32} eli 4 294 967 296. Näistäkin osa on varattu erikoistarkoituksiin kuten yksityisiin verkkoihin, jolloin julkisesti käytössä olevien osoitteiden määrä ei ole näinkään suuri. Internetiin liitettävien koneiden määrä kasvaa jatkuvasti, mutta osoitteita on rajallinen määrä. Tähän haasteeseen on kehitetty IPv6-osoitteistus. IPv6 käyttää 128-bittisiä osoitteita, jolloin osoitteita voi olla teoriassa 2^{128}. Tämä on huikea määrä verrattuna IPv4-osoiteavaruuteen, mutta IPv6 ei ole siltikään saavuttanut suurta suosiota vielä, vaan suositumpaa on ollut antaa IPv4-teknologialle tekohengitystä useilla eri menetelmillä kuten tässä luvussa esiteltävällä osoitteenmuunnoksella.

Osoitteenmuunnos eli lyhyemmin NAT (engl. Network Address Translation) on menetelmä, jolla piilotetaan suurempi osoiteavaruus yhden tai useamman julkisen IP-osoitteen taakse. Osoitteenmuunnoksessa on hieman sama idea kuin välityspalvelimessa, koska se tarjoaa turvallisuutta piilottaen julkiselta Internetiltä sisäverkon koneet. Osoitteenmuunnos on kuitenkin sallivampi liikennöinnin suhteen, jos niin halutaan. Välityspalvelin on aina keskittynyt jonkin sovelluksen toteuttamiseen, mutta osoitteenmuunnos tehdään verkkokerroksella reitittimessä, ja liikennöinti voi olla kaksisuuntaista lähteen ja kohteen välillä. Osoitteenmuunnokset voidaan jakaa neljään eri tyyppiin, joita ovat staattinen osoitteenmuunnos, dynaaminen osoitteenmuunnos, porttimuunnos ja käänteinen osoitteenmuunnos. Kaikissa tyypeissä yhteistä on se, että reitittimen sisäverkon puolella käytetään yksityiseksi määriteltyä IP-osoiteavaruutta, joka mainittiin IPv4:n yhteydessä, ja ulkoverkon puolella voidaan käyttää mitä tahansa julkista IP-osoitetta. Yksityisiä IP-osoiteavaruuksia ovat 10.0.0.0/8, 172.16.0.0/12 ja 192.168.0.0/16. Merkintä 10.0.0.0/8 tarkoittaa osoitteita 10.0.0.0 – 10.255.255.255, jolloin niitä on yhteensä 16 777 216. Merkintä 172.16.0.0/16 tarjoaa avaruuden väliltä 172.16.0.0 – 172.31.255.255, jolloin osoitteita on käytettävissä 1 048 576. Vastaavasti merkintä 192.168.0.0/16 tarkoittaa osoitteita väliltä 192.168.0.0 – 192.168.255.255, jolloin avaruus on kooltaan 65 536 osoitetta. Mikä tahansa näistä yksityisistä osoiteavaruuksista voidaan osoitteenmuunnoksen avulla sijoittaa periaatteessa jokaisen julkisen IP-osoitteen taakse, jolloin IPv4:n osoiteavaruus laajenee räjähdysmäisesti. [29]



Kuva 10: IP-osoitteiden uudelleenkäyttö osoitteenmuunnoksen avulla

Kuva 10 selventää osoitteenmuunnoksen ideaa. Verkot 1 ja 3 käyttävät samaa yksityisen verkon osoiteavaruutta, ja verkot 2 ja 4 käyttävät kahta muuta mainittua yksityistä osoiteavaruutta. Verkot 1 ja 3 käyttävät osittain samoja IP-osoitteita, mutta tämä on täysin mahdollista, koska koneet eivät liikennöi julkisessa Internetissä omalla IP-osoitteellaan vaan reitittimen julkisella IP-osoitteella. Otetaan esimerkiksi tilanne, jossa verkosta 1 ja 3 saman IP-osoitteen (192.168.0.2) omaavat koneet hakevat verkkosivun WWW-palvelimelta (81.22.244.45). Verkon 1 kone ottaa yhteyden WWW-palvelimeen, jolloin reititin vaihtaa lähdeosoitteeksi omansa eli 84.239.237.226. Samoin verkon 3 kone ottaa yhteyden WWW-palvelimeen, jolloin sen verkon reititin puolestaan vaihtaa lähdeosoitteeksi 84.239.237.227. Näin WWW-palvelin näkee, että yhteydet tulevat kahdesta eri julkisesta IP-osoitteesta. Samasta IP-osoitteesta tulevat pyynnöt eivät sinänsä ole ongelma WWW-palvelimelle, mutta jos osoitteenmuunnosta ei tehtäisi, WWW-palvelin lähettäisi molempien vastaukset samaan osoitteeseen, jolloin Internetin reitittimet eivät osaisi kuljettaa vastauksia oikeille koneille, koska tässä tapauksessa osoite on moniselitteinen. Tästä syystä on tärkeää, että Internetin julkiset IP-osoitteet ovat yksiselitteisiä. Toisin sanoen on vaatimus, että yksi IP-osoite johtaa vain yhteen Internetiin kytkettyyn koneeseen. Internetin reitittimet eivät edes suostu välittämään paketteja, joiden kohdeosoite kuuluu yksityiseen osoiteavaruuteen. Poikkeuksen muodostavat tietenkin reunareitittimet, jotka tekevät osoitemuunnosta, mutta samassa lähiverkossakaan ei ole sallittua olla moniselitteisiä osoitteita. Reunareitittimet ovat tilallisia, mikä tarkoittaa sitä, että kun kone avaa yhteyden lähiverkosta julkiseen Internetiin, reititin laittaa

muistiin tämän yhteyden, jotta se osaa välittää vastauspaketit takaisin oikealle koneelle. Käytännössä tämä tarkoittaa sitä, että kun vastauspaketti tulee reitittimelle, reititin vaihtaa jälleen kohdeosoitekentästä oman IP-osoitteensa lähiverkon koneen IP-osoitteeksi, jolloin lähiverkosta liikennöivä kone ei edes huomaa, että osoitteita on reitin varrella vaihdeltu. Luonnollisesti yksiselitteisyysvaatimus estää yhteydenmuodostuksen yksityisestä verkosta toiseen. Yhteydenmuodostus ei muutenkaan onnistu normaalitilanteessa Internetistä yksityiseen verkkoon, koska yksityisen verkon IP-osoitteet eivät edelleenkään ole sallittuja kohteita. Tämän takia onkin kehitetty eri tyyppisiä osoitteenmuunnoksia, jotta myös yksityiseen osoiteavaruuteen on mahdollista perustaa palvelimia, jotka ovat julkisesti saavutettavissa.

Taulukko 2: Staattinen osoitteenmuunnos.

Sisäverkon IP-osoite	**Ulkoverkon IP-osoite**
192.168.0.1	84.239.237.100
192.168.0.2	84.239.237.101
192.168.0.3	84.239.237.102
. . .	. . .
192.168.0.15	84.239.237.116
192.168.0.16	84.239.237.117

Staattinen osoitteenmuunnos eli static NAT toimii siten, että reitittimessä jokaiselle sisäverkon IP-osoitteelle määritellään järjestyksessä vastaava ulkoverkon IP-osoite. Taulukosta 2 voidaan nähdä, miten sisäverkon ja ulkoverkon IP-osoitteet kuvautuvat keskenään. Näin ei kylläkään säästetä osoiteavaruutta, koska julkisia IP-osoitteita kuluu yhtä paljon kuin lähiverkossa on koneita. Tässä etuna on se, että yhteydenotto ulkoverkosta päin lähiverkon koneeseen on mahdollista, mutta minkäänlaista suojaa ei toisaalta ole tarjolla, koska kone näkyy julkisesti Internetissä. [30, s. 4]

Taulukko 3: Dynaaminen osoitteenmuunnos.

Sisäverkon IP-osoite	**Ulkoverkon IP-osoite**
192.168.0.5	84.239.237.100
192.168.0.46	84.239.237.101
192.168.0.87	84.239.237.102
. . .	. . .
192.168.0.137	84.239.237.116
192.168.0.244	84.239.237.117

Dynaaminen osoitteenmuunnos eli dynamic NAT säästää julkisia IP-osoitteita, koska reitittimeen voidaan konfiguroida tietyn suuruinen varanto IP-osoitteita, jois-

ta jaetaan tarvitsijoille yhteyksiä. Taulukosta 3 nähdään osoitteidenjaon periaate. Lähiverkossa voi olla paljonkin enemmän koneita kuin varannossa on julkisia IP-osoitteita. Kun on tarve ottaa yhteys julkiseen Internetiin, käytetään dynaamisesti jotain vapaana olevaa julkista IP-osoitetta liikennöintiin. Jos yhtään julkista IP-osoitetta ei ole vapaana, ei yhteyttä julkiseen Internetiin voida muodostaa. Tässä tapauksessa ulkoverkosta ei ole mahdollista ottaa yhteyttä sisäverkon koneeseen, koska mikään varannon IP-osoitteista ei johda millekään tietylle koneelle lähiverkossa. Tästä on luonnollisesti se hyöty, että lähiverkon koneet ovat turvassa suorilta tietomurtoyrityksiltä. [30, s. 4]

Taulukko 4: Porttimuunnos.

Sisäverkon IP-osoite ja portti	**Ulkoverkon IP-osoite ja portti**
192.168.0.5:2111	84.239.237.100:21
192.168.0.46:2222	84.239.237.100:22
192.168.0.87:8080	84.239.237.100:80

Porttimuunnoksen eli PAT:n (engl. Port Address Translation) tai NAPT:n (engl. Network Address Port Translation) idea nähdään taulukosta 4. Tällä tekniikalla saadaan todellinen osoiteavaruuden säästö, koska yhden IP-osoitteen eri portit voidaan ohjata mielivaltaisesti eri lähiverkon koneille. Porttimuunnoksen avulla on mahdollista ottaa yhteyttä ulkoverkosta sisäverkon koneille, joissa on tarjolla palveluita. Nämä palvelut näkyvät ulospäin siten, että ne olisivat kaikki samassa koneessa, vaikka jokainen palvelu voi todellisuudessa sijaita fyysisesti eri koneella. Lähiverkon kaikki koneet voivat liikennöidä vapaasti yhden julkisen IP-osoitteen takaa. Porttimuunnos kuuluu normaalien kotireitittimien ominaisuuksiin, koska monet Internet-palveluntarjoajat tarjoavat vain yhden julkisen IP-osoitteen asiakasta kohden, joten ainoaksi vaihtoehdoksi jää muodostaa oma lähiverkko kotiin, jossa kaikki lähiverkon koneet liikennöivät saman julkisen IP-osoitteen kautta. Jos asiakas haluaa tarjota palveluita julkiseen Internetiin, palvelut on helppo ohjata oikeille lähiverkon koneille porttimuunnoksen avulla. Taulukon 4 esimerkkiin on poimittu joitakin yleisimpiä palveluita kuten SSH, FTP ja WWW. SSH on palvelu, jota tässäkin työssä tullaan myöhemmin hyödyntämään huoltoyhteyden muodostukseen. [30, s. 4]

Taulukko 5: Käänteinen osoitteenmuunnos.

Sisäverkon IP-osoite ja portti	**Ulkoverkon IP-osoite ja portti**
192.168.0.5:80	
192.168.0.46:80	84.239.237.100:80
192.168.0.87:80	

Käänteinen osoitteenmuunnos eli reverse NAT kääntää nimensä mukaisesti osoitteenmuunnoksen idean päälaelleen. Kun edelliset osoitteenmuunnoksen tyypit ovat

perustuneet siihen, että sisäverkon IP-osoitteita vaihdetaan ennen pakettien lähettämistä Internetiin, tehdäänkin tässä muunnos ulkoverkosta tulevalle liikenteelle. Lähiverkon koneista voidaan muodostaa palveluryväs, jonne liikennettä ohjataan siten, että se jakaantuu tasaisesti kaikkien koneiden kesken. Tällöin reititin toimii kuormanjakajana. Kuormanjakoa joudutaan tekemään silloin, kun esimerkiksi ylläpidetään suosittua WWW-palvelinta, jonne saapuu palvelupyyntöjä niin paljon, että yksi kone ei ehdi käsitellä kaikkia pyyntöjä kohtuullisessa ajassa. Käänteisen osoitteenmuunnoksen avulla kuorma jakaantuu tasaisesti, ja palvelun vasteajat pysyvät nopeina. WWW-palvelu näkyy käyttäjille jälleen vain yhtenä koneena. Tämä voidaan todeta taulukosta 5. [31]

Osoitteenmuunnoksesta on paljon hyötyä, koska se auttaa IPv4-osoitteiden vähyyteen, tarjoaa hieman suojaa suorilta hyökkäyksiltä Internetistä, ja sillä voidaan tasoittaa kuormaa useammalle koneelle, mutta on myös protokollia, joille osoitteenmuunnos ei sovi. Esimerkiksi tiedonsiirtoon tarkoitettu FTP protokolla ei osaa toimia aktiivisessa tilassa osoitteenmuunnoksen kanssa. Kun käyttäjä muodostaa lähiverkosta NAT:n takaa yhteyden johonkin Internetissä olevaan FTP palvelimeen, käyttäjä voi selailla tätä yhteyttä käyttäen palvelimella tarjolla olevia tiedostoja. Kun käyttäjä haluaa siirtää jonkin palvelimella olevan tiedoston itselleen, siirto ei tapahdu saman yhteyden välityksellä, vaan palvelin ottaa uuden yhteyden asiakkaaseen lähettääkseen tiedoston. Tämä ei tietenkään onnistu, sillä palvelin ottaa yhteyttä reitittimeen, koska luulee yhteyden tulevan reitittimen osoitteesta muunnetun osoitteen perusteella. Osoitteenmuunnos on myös yksi syy, minkä takia Oliotalon päätelaitteet ovat aktiivinen osapuoli yhteydenmuodostuksessa. Osoitteenmuunnoksen takia ei myöskään pystytä luomaan suoraan huoltoyhteyttä päätelaitteelle, vaan tässäkin tapauksessa päätelaitteen on itse otettava yhteys, vaikkakin se käskettäisiin palvelimelta käsin. Vaikka porttimuunnoksen tai staattisen osoitteenmuunnoksen avulla voitaisiin saada päätelaite näkymään julkisessa Internetissä, se ei ole kuitenkaan vaihtoehto, koska Oliotalo ei pääse useinkaan vaikuttamaan niiden verkkojen hallintoon, jonne asiakas päätelaitteet asentaa. Tästä syystä ratkaisun tulee olla sellainen, että se toimii varmasti verkon konfiguraatiosta riippumatta.

4.5 Yhteenveto

Luku aloitettiin tutustumalla Internetin OSI-mallin mukaiseen rakenteeseen ja liikennöintiin. Nopeasti havaittiin, että Internetissä on teknologioita, jotka häiritsevät vapaata tiedon liikkumista. Opittiin, että organisaatioissa on käytössä palomuureja, joilla rajoitetaan liikennöintiä joko siitä syystä, että halutaan pitää hyökkääjät loitolla organisaation verkosta, tai estää omia työntekijöitä liikennöimästä haitallisten tai työtehtävien kannalta tarpeettomien koneiden kanssa. Organisaatiot käyttävät myös välityspalvelimia, joilla voidaan suodattaa liikennettä monipuolisesti ja vähentää verkkoliikennettä. Lopuksi kerrottiin IPv4-osoitteiden vähyydestä, johon ratkaisuna on kehitetty osoitteenmuunnos. Osoitteenmuunnos tarjoaa turvaa lähiverkon koneille mutta haittaa joitakin sellaisia sovelluksia, jotka on suunniteltu toimimaan ainoastaan julkisella IP-osoitteella.

Näiden häiritsevien elementtien takia Oliotalo on suunnitellut oman alustansa

siten, että päätelaite on aktiivinen osapuoli yhteyden muodostamisessa, koska tällä tavoin voidaan välttää osa sudenkuopista. Tässä luvussa esitellyt häiritsevät elementit tulee pitää mielessä, kun lähdetään suunnittelemaan huoltoyhteyttä.

5 SSH - askel kohti ratkaisua

Tämän työn päämääränä on löytää toimiva ratkaisu huoltoyhteyden muodostamiseksi päätelaitteisiin. SSH eli Secure Shell on alunperin Tatu Ylösen kehittämä tietoturvallinen protokolla kahden koneen väliseen tiedonsiirtoon. Se sopii hyvin osaratkaisuksi, koska se tarjoaa etäkomentokehoteyhteyden, joka on samalla salattu. Ennen SSH:n kehittämistä etäkomentokehote luotiin käyttäen `telnet-`, `rsh-` ja `rlogin-`ohjelmistoja, mutta ne eivät tarjonneet liikenteen salausta, koska Internetin alkuaikoina verkot olivat vielä suljetumpia ja luotettavia, mutta nykyään Internetissä on paljon toimijoita, joista kaikki eivät ole rehellisiä, joten liikenteen salaus on oikeastaan minimivaatimus etäyhteyttä luodessa, koska samalla siirretään käyttäjätunnus ja salasana, joiden joutuminen vääriin käsiin antaa hyökkääjälle pääsyn palvelimelle. SSH tarjoaa samat ominaisuudet kuin `telnet`, `rsh` ja `rlogin` mutta salattuna, joten valinta tähän tarkoitukseen on helppo. Mitä siis vaaditaan, että etäkomentokehote saadaan luotua SSH:n avulla kahden koneen välille? SSH toimii jo tutuksi tulleella asiakas–palvelin-periaatteella, eli palvelin kuuntelee tiettyä porttia, jonne asiakas ottaa yhteyden. Yhteydenmuodostuksen jälkeen etäkonetta voidaan komentaa komentoriviltä kuten paikallistakin konetta.

Tarvittavat SSH-ohjelmistot löytyvät yleisimmistä Linux-jakeluista. Palvelinkoneeseen tarvitsee asentaa `sshd` eli SSH-demoni, jota suoritetaan taustaprosessina. `sshd` kuuntelee oletuksena yhteyksiä portissa `22` mutta on konfiguroitavissa käyttämään jotain muutakin porttia. Palvelimella tulee olla luotuna käyttäjä, jolla on oikeus kirjautua järjestelmään. Asiakaskoneelle tulee asentaa `ssh`-ohjelma, jolla etäkirjautuminen tehdään. Perustapauksessa kirjautuminen palvelimelle tapahtuu komentamalla asiakaskoneessa

`ssh remoteserver.host.com` ,

jossa `remoteserver.host.com` on palvelimen osoite. Osoite voi olla myös numeerisessa muodossa. Tämän jälkeen palvelin kysyy salasanaa, ja jos se kirjoitetaan oikein, komentokehoteyhteys palvelimelle on luotu. Jos palvelimelle halutaan kirjautua eri käyttäjätunnuksella kuin sillä hetkellä on käytössä asiakaskoneessa ja käyttää jotain muuta porttia kuin oletusta, yhteys tulee luoda komennolla

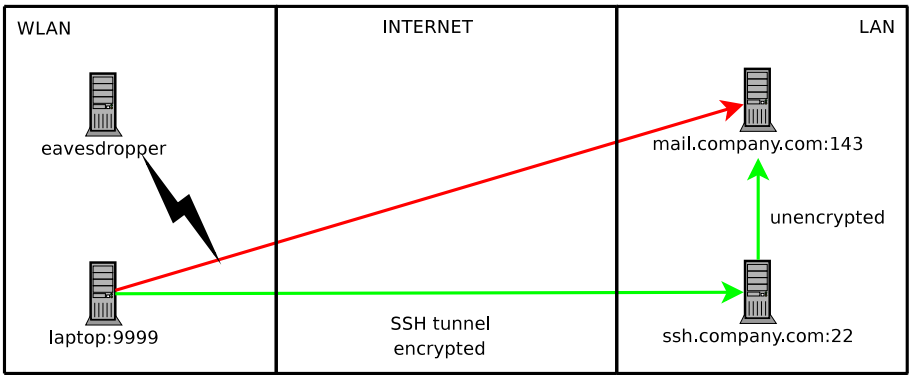
`ssh -l user -p port remoteserver.host.com` ,

jossa `user` on kirjautuva käyttäjä ja `port` on käytettävä portti. Normaalitapauksessa komentokehoteyhteyden luominen etäkoneelle ei ole tämän vaikeampaa, mutta tämän työn tapauksessa se ei ole näin helppoa, koska päätelaite, jonne yhteys pitäisi ottaa, ei ole yleensä kiinni julkisessa Internetissä vaan palomuurin tai NAT:n takana. Tällöin tarvitsee luoda käänteinen tunneli.

5.1 Käänteinen tunneli

SSH:n ominaisuuksiin kuuluu tunnelointi. Tunnelointi tarkoittaa sitä, että `ssh`-ohjelma on mahdollista asettaa liikenteen välittäjäksi ja salaajaksi. Otetaan esimerkkinä tilanne, jossa käyttäjä on liittynyt kannettavalla tietokoneella hotellin salaamattomaan WLAN-verkkoon ja haluaa lukea sähköpostit yrityksen sähköpostipalvelimelta mutta on huolissaan salakuuntelijoista. Yrityksellä sattuu olemaan säh-

köpostipalvelimen kanssa samassa sisäverkossa kone, johon käyttäjällä on pääsy ja SSH asennettuna. Käyttäjä voi luoda salatun SSH-tunnelin tähän koneeseen, jota kautta liikenne voidaan ohjata sähköpostipalvelimelle. Kaiken tämän tekeminen onnistuu hotellin WLAN-verkosta turvallisesti.



Kuva 11: SSH-tunnelointi

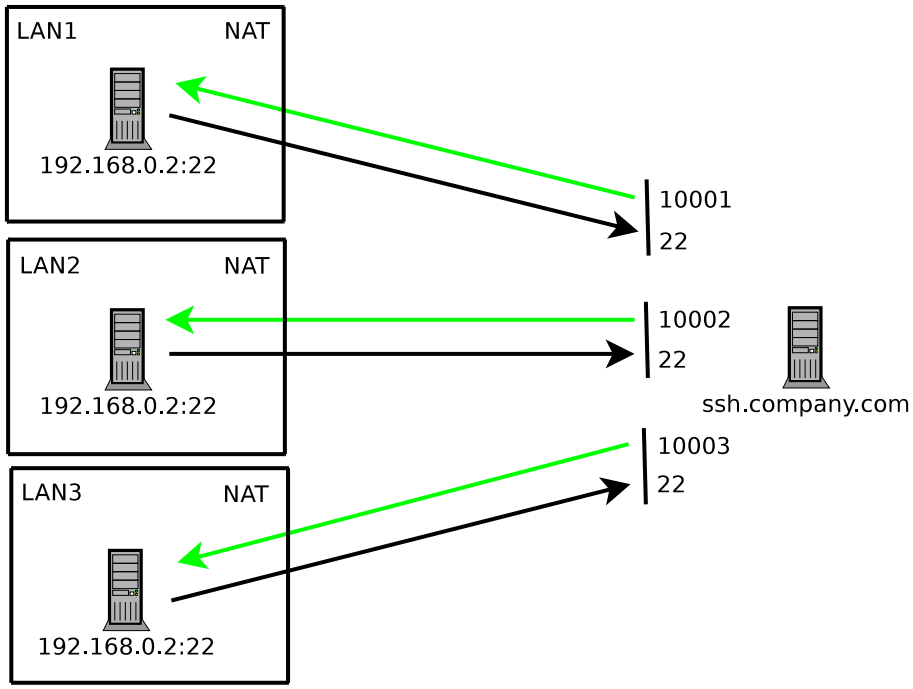
Kuvassa 11 nähdään, kuinka käyttäjä on muodostanut yhteyden hotellin WLAN-verkosta kannettavalla tietokoneella `laptop` yrityksensä sähköpostipalvelimelle `mail.company.com` porttiin `143`. Tämä yhteys näkyy kuvassa punaisella viivalla. Salakuuntelija `eavesdropper` pystyy lukemaan kaiken liikenteen vapaasti, koska WLAN-verkko on salaamaton. Estääkseen salakuuntelun, käyttäjä voi muodostaa salatun SSH-tunnelin koneelle `ssh.company.com`, joka ohjaa edelleen liikenteen sähköpostipalvelimelle. Käyttäjä voi komentaa omalla kannettavalla tietokoneellaan

```
ssh -L 9999:mail.company.com:143 -l user ssh.company.com
```

, jonka jälkeen käyttäjä `user` antaa salasanan koneelle `ssh.company.com`. Komento laittaa koneen `laptop` kuuntelemaan paikallisesti porttia `9999` ja käskee konetta `ssh.company.com` ohjaamaan kaiken liikenteen koneelle `mail.company.com` porttiin `143`. Nyt käynnissä on salattu tunneli koneiden `laptop` ja `ssh.company.com` välillä. Liikenne on salaamatonta koneiden `ssh.company.com` ja `mail.company.com` välillä, mutta se ei haittaa, koska liikenne kulkee turvallisesti yrityksen omassa lähiverkossa. Käyttäjä on luonut suoran yhteyden oman koneen portista `9999` sähköpostipalvelimen porttiin `143`, josta kriittisin osuus on salattu. Tämä ketju näkyy kuvassa vihreällä viivalla. Seuraavaksi käyttäjän pitää vain vaihtaa sähköpostiohjelman asetuksista sähköpostipalvelimeksi `localhost` ja portiksi `9999`, koska nyt sähköpostipalvelin näyttäisi olevan omassa koneessa. `localhost` on niin sanottu loopback-laite eli alias oman koneen IP-osoitteelle, joka numeerisesti on `127.0.0.1`. Näin käyttäjä pääsee lukemaan sähköpostin turvallisesti, vaikka käyttäisikin turvattomia verkkoja välissä.

Palataan alkuperäiseen ongelmaan, ja katsotaan, miten tehdään käänteinen tunneli. Käänteisen tunnelin avulla päästään kirjautumaan koneille, jotka ovat NAT:n takana. Otetaan esimerkkinä tilanne, jossa kolmessa eri verkossa on kone, jonne täytyy päästä kirjautumaan etänä.

Kuvassa 12 vasemmalla on kolme eri verkkoa, joissa kaikissa käytetään osoit-



Kuva 12: Käänteinen SSH-tunneli

teenmuunnosta, eikä yksikään koneista näy julkiseen Internetiin. Esimerkin vuoksi koneille on sattunut vielä sama IP-osoite, mutta se ei ole tässä tapauksessa ongelma, kuten tullaan näkemään. Näihin kolmeen koneeseen on asennettu sekä `ssh`- että `sshd`-ohjelmistot, ja ne kuuntelevat porttia 22. Oikealla nähdään kone `ssh.company.com`, joka toimii tässä tapauksessa keskuskoneena, jonne NAT:n takana olevat koneet muodostavat käänteisen tunnelin. Käänteisen tunnelin avulla koneelta `ssh.company.com` voidaan kirjautua keskitetysti NAT:n takana oleviin koneisiin. Luodakseen käänteisen tunnelin, LAN:eissa sijaitsevat koneet komentavat seuraavasti:

LAN1: `ssh -R 10001:localhost:22 -l user ssh.company.com`
LAN2: `ssh -R 10002:localhost:22 -l user ssh.company.com`
LAN3: `ssh -R 10003:localhost:22 -l user ssh.company.com` .

Käänteinen tunneli muodostuu salasanan syötön jälkeen. Ensimmäinen komento käskee koneen `ssh.company.com` kuunnella yhteyksiä portista `10001` ja ohjata yhteydenotot tunnelin muodostajan porttiin 22 käyttäen yhteyttä, jolla tunnelinavauspyyntö toteutettiin. Muut komennot tekevät samoin, mutta ne käskevät kuunnella eri porttia. Nyt koneelta `ssh.company.com` voidaan ottaa komentokehoteyhteys eri LAN:eihin seuraavasti:

LAN1: `ssh -l user -p 10001 localhost`
LAN2: `ssh -l user -p 10002 localhost`
LAN3: `ssh -l user -p 10003 localhost` .

Ensimmäinen komento on tuttua muotoa eli käyttäjä `user` kirjautuu saman koneen `localhost` porttiin `10001`. Annettuaan salasanan, käyttäjällä on toimiva komento-

kehoteyhteys LAN1:n koneeseen. Vastaavasti toimii yhteydenotto muihin LAN:eihin, mutta käytetään vain eri porttia. Kirjautuminen on nyt mahdollista, koska varsinainen yhteys on avattu LAN:sta päin, mikä on sallittua, mutta SSH:n avulla yhteys onkin käännetty niin päin, että yhteyden toisesta päästä onkin mahdollista tehdä kirjautuminen komentokehotteeseen. Reitittimien näkökulmasta tämä on hyväksyttävää kaksisuuntaista liikennöintiä, joka on aloitettu oikeaoppisesti LAN:sta päin.

5.2 Julkiseen avaimeen perustuva tunnistautuminen

Nyt ollaan jo lähellä toimivaa tunnelointiratkaisua mutta ei aivan tavoitteessa. Ongelmaksi muodostuu salasanan syöttäminen tunnelin luomisen yhteydessä, koska kukaan ei ole syöttämässä salasanaa päätelaitteelle, kun se suorittaa itsenäisesti valvontaa jossain mahdollisesti hankalakulkuisessa paikassa. Muutenkin huoltoyhteyden luomisen yhtenä motiivina oli, ettei kenenkään tarvitse lähteä käymään fyysisesti paikan päällä, vaan huoltoyhteys tulee pystyä avaamaan keskitetysti yhdestä paikasta. Onneksi SSH:n ainoa todentamismenetelmä ei ole salasana, vaan todentamisessa voidaan käyttää julkiseen avaimeen perustuvaa tunnistautumista.

Julkiseen avaimeen perustuva tunnistautuminen toimii siten, että kirjautuja luo itselleen sekä julkisen että salaisen avaimen, jotka ovat keskenään avainpari. Avainpari toimii puolestaan siten, että julkisella avaimella salattu tieto voidaan purkaa vain salaisella avaimella. Avainparin luomiseen tarvitaan `ssh-keygen`-ohjelmisto. RSA algoritmiin perustuva avain voidaan luoda komennolla

`ssh-keygen -t rsa` .

Avaimet tallentuvat käyttäjän `user` kotihakemistoon seuraavasti:

julkinen avain `/home/user/.ssh/id_rsa.pub` ja

salainen avain `/home/user/.ssh/id_rsa` .

Näistä kahdesta julkinen avain tulee kopioida koneeseen, jonne on tarkoitus päästä kirjautumaan ilman salasanaa. Käytetään esimerkkinä tuttua konetta `ssh.company.com`. Julkinen avain lisätään käyttäjän `user` kotihakemistossa olevaan tiedostoon

`/home/user/.ssh/authorized_keys`

koneella `ssh.company.com`. Jos tiedostoa ei vielä ole olemassa, se täytyy luoda käsin. Nyt koneelle `ssh.company.com` voi kirjautua ilman salasanan syöttämistä. Komento on vanha tuttu eli

`ssh -l user ssh.company.com` .

Julkisen avaimen käyttäminen tunnistautumisessa on turvallista, koska missään vaiheessa ei tarvitse lähetellä salasanaa. Salainen avain tulee luonnollisesti pitää salaisena, koska jos salainen avain joutuu hyökkääjän käsiin, turvallisuusketju on tällöin murrettu, ja hyökkääjällä on yhtäläinen pääsy kohdekoneelle kuin oikeallakin käyttäjällä.

5.3 Dropbear-ohjelmisto

Koska Oliotalon käyttämä päätelaite on sulautettu tietokone, halutaan ratkaisun pohjana käyttää mahdollisimman kevyttä SSH-ohjelmistoa. Tähän tarkoitukseen

on olemassa Dropbear niminen ohjelmisto. Sen keveys perustuu siihen, että joitakin ohjelmiston osia on jätetty pois verrattuna esimerkiksi OpenSSH-ohjelmistoon, joka on asennettavissa Linux-jakeluihin jonkin pakettihallinnan kautta. Dropbear-ohjelmistosta on karsittu esimerkiksi tuki vanhemmalle SSH1-protokollalle, jota ei ole suositeltava edes käyttää, ja puutteena on `scp`-ohjelma, jolla on mahdollista siirtää tiedostoja koneelta toiselle. Näitä ominaisuuksia ei tarvita tämän työn toteuttamiseen, koska ei ole tarvetta kopioida tiedostoja SSH:n välityksellä, eikä työssä haluta käyttää vanhentuneita ja tietoturvattomia protokollia, vaan nykyään käytössä oleva SSH2 on hyvä vaihtoehto hyvän tietoturvansa ansiosta. Dropbear-ohjelmiston lähdekoodipaketti on ladattavissa Internetistä. [32]

Oliotalon pöytätietokoneiden suoritinarkkitehtuuri on x86, joten Dropbear-ohjelmisto täytyy ristiinkääntää ARM-arkkitehtuurille, koska tämä on Platypuksen suoritinarkkitehtuuri, kuten todettiin aiemmin. Kääntäminen on helpointa tehdä Scratchbox ristiinkääntöympäristöllä. Tässä työssä ei keskitytä ristiinkääntämiseen sen enempää, joten todetaan vain, että kääntäminen tuottaa lähdekooditiedostoista ajettavia ohjelmia. Käännöksen tuloksena syntyy `dbclient`-, `dropbear`- ja `dropbearkey`-ohjelmistot. `dbclient` vastaa `ssh`-ohjelmistoa eli asiakasohjelmistoa. `dropbear` vastaa `sshd`-ohjelmistoa eli palvelinohjelmistoa. `dropbearkey` vastaa `ssh-keygen`-ohjelmistoa eli avainparin luontiin tarvittavaa ohjelmistoa. Avainlukupari saadaan luotua komennolla

```
dropbearkey -t rsa .
```

Palvelin voidaan käynnistää yksinkertaisesti taustaprosessiksi komennolla

```
dropbear & .
```

`dbclient`-ohjelmalla on tarkoitus saada käänteinen tunneli luotua, joten katsotaan, millä parametreilla se pitää käynnistää. Tutkimalla ohjelman dokumentaatiota, tunneli päädyttiin muodostamaan komennolla

```
dbclient -i/etc/dropbear/dropbear_rsa_host_key -luser -N -f -y
-K 60 -R port:127.0.0.1:22 server .
```

Vipu `-i` kertoo salaisen avaimen sijainnin tiedostojärjestelmässä, mitä tarvitaan julkisen avaimen tunnistautumisessa. Tätä salaista avainta vastaava julkinen avain tulee sijoittaa koneeseen, mihin yhteys otetaan, kuten aiemmin käytiin läpi. Vipu `-l` kertoo käyttäjän. Vipu `-N` ilmoittaa, että komentokehotetta ei tarvitse turhaan avata kohdekoneeseen, koska sitä ei ole kukaan käyttämässä. Tämä ei siis estä komentokehotteen muodostumista toiseen suuntaan, kun kohdekoneesta kirjaudutaan tunnelia pitkin Platypukselle. Vipu `-f` käskee suorittaa ohjelman taustaprosessina tunnistautumisen jälkeen. Vipu `-y` sallii tuntemattomien isäntäavaimien hyväksynnän ilman käyttäjän syötettä, koska käyttäjä ei ole edelleenkään antamassa mitään syötettä, joten tunnelin on muodostuttava ilman kyselyjä. Vivulla `-K` kerrotaan ohjelmalle aikaväli sekunneissa, jonka välein tunneliin on lähetettävä niin sanottu keepalive-viesti, jolla yhteys pidetään aktiivisena, koska reitittimet saattavat aikakatkaista joutilaita yhteyksiä. Koska reitittimien aikakatkaisun arvoja ei voida tietää, arvoksi on valittu 60 sekuntia eli minuutti, jonka pitäisi olla tarpeeksi tiheä aikaväli. Mikäli reitittimien aikakatkaisu olisi säädetty tätä nopeammaksi, se haittaisi jo normaalia Internetin käyttöä. Vipu `-R` on varsinainen käänteisen tunnelin muodostusvipu, jolla kerrotaan, että kohdekoneen tulee kuunnella porttia `port` osoitteessa `127.0.0.1`,

joka tunnetaan myös aliaksella `localhost`. Viimeinen parametri `server` ilmoittaa kohdekoneen IP-osoitteen.

5.4 Yhteenveto

Luku aloitettiin tutkimalla SSH-ohjelmistojen soveltuvuutta huoltoyhteyden osaratkaisuksi. Valinta ei ollut vaikea, koska SSH tarjoaa kaiken, mitä huoltoyhteyden muodostamiseen vaaditaan. Seuraavaksi tutustuttiin SSH ohjelmiston ominaisuuksiin ja komentoihin. Lopuksi ratkaisun toteutukseksi valittiin sulautetuille järjestelmille soveltuva kevyempi Dropbear-ohjelmisto. Dropbear-ohjelmistosta opeteltiin tarvittavat komennot, joilla käänteinen tunneli voidaan muodostaa.

Nyt on kaikki tarvittava tieto, jotta huoltoyhteys voidaan muodostaa SSH:n tarjoamaa käänteistä tunnelia pitkin. Seuraavaksi lähdetään rakentamaan kokonaisratkaisua, jotta huoltoyhteys saadaan asennettua ja aktiiviseksi palvelimelta käsin.

6 Toteutus

Tässä luvussa keskitytään rakentamaan alkuperäiselle ongelmalle ratkaisu. Ratkaisussa käytetään hyväksi aiemmin opittuja asioita, mutta mietitään tavoitteen saavuttamiseksi myös uusia. Aluksi mietitään, miten toteutus tulisi tehdä. Seuraavaksi käydään läpi, miten SSH-ratkaisu paketoidaan toimivaksi kokonaisuudeksi, jotta sitä voi käyttää päätelaitteessa tunnelin aikaansaamiseksi. Tämä vaatii apuskriptien luomista ja M2M-alustan muokkaamista siten, että SSH:n asennus voidaan tehdä etänä. Ratkaisun toimivuutta tullaan luonnollisesti testaamaan. Tämän jälkeen on vielä yksi ongelma selätettävänä. Tämä ongelma on lähiverkkojen liikennettä rajoittava palomuuri. Ongelmaan tullaan toteuttamaan palomuurin kiertävä ratkaisu. Kun kokonaisratkaisu on valmis, tehdään kenttätestit, mikä käytännössä tarkoittaa sitä, että tuotannossa oleviin päätelaitteisiin tullaan tekemään Linux-ytimen päivitys. Jos ratkaisu toimii, tavoite on saavutettu.

6.1 Rakenne

Mietitään aluksi, millainen huoltoyhteyden muodostuksen ja hallinnan rakenne tulisi olla. Listataan asiat toimenpiteistä, joita käyttäjä tekee käyttäessään huoltoyhteyttä.

Käyttäjä

1. asentaa SSH-ohjelmiston.
2. käskee päätelaitetta muodostamaan käänteisen tunnelin.
3. kirjautuu päätelaitteelle.
4. tekee huoltotoimenpiteet.
5. kirjautuu ulos päätelaitteelta.
6. käskee päätelaitetta lopettamaan käänteisen tunnelin.

Kohta 1. tarvitsee luonnollisesti tehdä vain kerran samalle päätelaitteelle. Loput kohdat tehdään aina, kun käänteinen tunneli halutaan muodostaa. SSH-ohjelmiston asennusta varten tarvitsee toteuttaa palvelimeen sivu, josta tarvittavat tiedostot voidaan syöttää lähetettäväksi. Sparrow-protokollaan on jo aiemmin toteutettu tiedostonsiirtomahdollisuus palvelimelta päätelaitteelle, joten tässä käytetään hyväksi sitä. Kohta 2. on helpoin toteuttaa siten, että WWW-palvelimelle asetetaan päätelaitteen sarjanumerolla hakemisto, jota päätelaite käy tarkastelemassa tasaisin väliajoin. Palvelimella olevasta hakemistosta löytyy tieto, mihin porttiin päätelaitteen tulee käskeä käänteinen tunneli muodostamaan. Kohta 3. tapahtuu jo opitun tavan mukaan ottaa yhteys käänteistä tunnelia pitkin päätelaitteelle. Kohta 4. voi olla mitä tahansa, mutta tässä työssä tavoitteena on saada Linux-ydin ja sen vaatimat ajurimoduulit päivitettyä. Kohta 5. on yksinkertainen eli kirjoitetaan komento `exit` komentoriville. Kohta 6. voidaan toteuttaa helposti vain poistamalla WWW-palvelimelta hakemisto, josta päätelaite hakee sarjanumeronsa perusteella tiedoston. Tämän mukaista rakennetta mukaillen lähdetään toteuttamaan ratkaisua.

6.2 Skriptien luonti ja SSH-ohjelmiston asennus

Tämän luvun tarkoituksena on miettiä, millaisia skriptejä tarvitaan, jotta SSH-ohjelmisto pystytään asentamaan etänä päätelaitteelle. Tavoitteena on tehdä sellainen toteutus, että päätelaitteen mittausohjelmistoa ei tarvitse päivittää SSH-ohjelmiston asennuksen yhteydessä. Toisin sanoen skripteistä tulee tehdä sellaisia, että ne voivat toimia itsenäisesti ilman, että mittausohjelmisto edes tietää niiden olemassaolosta.

Tutustutaan ensin hieman päätelaitteen tiedostojärjestelmään. 256 Megatavun NAND-muisti on jaettu kahteen 128 Megatavun osioon. Näistä ensimmäisellä osiolla on niin sanottu juuritiedostojärjestelmä, jonne on asennettu tarvittavat varusohjelmistot ja niiden tarvitsemat asennustiedostot. Toisella osiolla on kaikki mittausohjelmistoon liittyvät ohjelmistot, skriptit ja asetustiedostot. Tämä osio on liitetty juuritiedostojärjestelmän hakemistoon `/rw`. SSH-ohjelmisto on tarkoitus asentaa juuritiedostojärjestelmään yleisesti käytössä oleviin hakemistoihin muiden varusohjelmistojen tapaan, ja omat luotavat skriptit asennetaan hakemistoon `/rw`. Ohjelmistot ja skriptit voitaisiin asentaa minne tahansa, mutta selkeyden vuoksi on järkevää käyttää Oliotalon luomien ohjelmistojen sijoituspaikkana hakemistoa `/rw`.

Toinen asia, mitä tulee tietää mittausohjelmiston ja M2M-alustan toiminnasta, on yhteyksien hallinta. Mittausohjelmiston carrier-sm-tilakone pyrkii olemaan joko GPRS- tai Ethernet-yhteydessä. Toisen yhteystavan katketessa, yritetään toista yhteystapaa. Mittausohjelmisto ilmoittaa halunsa muodostaa GPRS-yhteys luomalla tiedostojärjestelmään tyhjän tiedoston nimeltä `/tmp/ppp-wanted`. Vastaavasti halu muodostaa Ethernet-yhteys tapahtuu luomalla tiedosto `/tmp/eth-wanted`. Näitä tiedostoja tarkkaillaan yhteydenmuodostusskripteistä, joita suoritetaan tasaisin väliajoin Linux-järjestelmissä yleisesti käytetystä cron-ajastuspalvelusta. Jos skriptit huomaavat, että halutaan muodostaa yhteys, skriptit luovat käskystä yhteyden, ja sen onnistuessa asettavat vastaavasti joko `/tmp/ppp-running` tai `/tmp/eth-running` tiedoston. Kun tällainen tiedosto ilmestyy tiedostojärjestelmään, se on merkki mittausohjelmistolle, että se voi yrittää luoda TCP-yhteyttä palvelimeen. Jos mittausohjelmisto haluaa lopettaa yhteyden, sen tarvitsee vain poistaa halua ilmoittava tiedosto tiedostojärjestelmästä, jolloin yhteysskriptit lopettavat yhteyden ja poistavat running-päätteisen tiedoston. Yhteyden katkeaminen voidaan myös huomata skriptien toimesta, jolloin running-päätteinen tiedosto poistetaan, ja se on merkki ohjelmistoille, että yhteyttä Internetiin ei ole olemassa. Tätä tietoa voidaan käyttää myös hyväksi SSH-tunnelin hallinnoimisessa.

Aloitetaan luomalla skripti `start-dropbear.sh`, jonka tarkoitus on hallinnoida tunnelin muodostamista. Skripti tullaan käynnistämään taustalle järjestelmän käynnistysskriptistä `/etc/init.d/rcS`, joten ylimmälle tasolle luodaan `while`-silmukka, jota suoritetaan ikuisesti nukkuen `timeout`-arvon verran jokaisella silmukan suorituskerralla. `timeout` on aluksi 60 sekuntia, jolloin yhteyksien valvonta on tarpeeksi nopeaa. Silmukan ensimmäisellä kerroksella valvotaan tiedostojen `/tmp/ppp-running` ja `/tmp/eth-running` olemassaoloa. Mikäli toinen tiedostoista ilmestyy järjestelmään, päätelaitteella on yhteys Internetiin. Seuraavalla kerroksella tutkitaan, onko tunnelin muodostamiseen tarvittavat asetukset haettu eli onko tiedosto `/tmp/re-`

moteshell.txt olemassa. Jos tiedostoa ei ole olemassa, se haetaan käyttäen wget-ohjelmaa komennolla

```
wget -O /tmp/remoteshell.txt
http://ssh.company.com/SERIALNR/remoteshell.txt.
```

Vipu -O käskee kirjoittamaan tiedoston /tmp/remoteshell.txt uudestaan, mikäli se on jo olemassa. Tämä on vain varmistuksena, ettei missään tapauksessa uusi haettu tieto jää tallentumatta. SERIALNR tilalle laitetaan aina kunkin laitteen sarjanumero, joka on luettavissa tiedostojärjestelmästä. Tiedoston remoteshell.txt sisältönä on yksinkertaisesti palvelin ja portti välilyönnillä eroteltuna. Tämän jälkeen palvelin ja portti seulotaan tiedostosta, käynnistetään ensin dropbear-palvelin ja sen jälkeen dbclient-asiakas aiemmin opituilla komennoilla. Jos tilanne pysyy ennallaan, silmukan seuraavilla kierroksilla ei tehdä mitään. Jos tiedostoa remoteshell.txt ei voida ladata, se on merkki siitä, että tunnelia ei haluta muodostaa, ja tällöin timeout-arvoksi asetetaan 600 sekuntia eli 10 minuuttia, koska tiedostoa ei ole järkevä yrittää hakea jatkuvasti mutta kuitenkin sen verran tiheästi, ettei yhteyden muodostuminen kestä kohtuuttoman pitkiä aikoja. Mikäli tiedosto saadaan haettua, mutta siitä ei saada luettua palvelinta ja porttia, tuhotaan tiedosto, jotta se voidaan hakea uudestaan seuraavalla kierroksella. Jos silmukan ylätasolla huomataan, ettei Internet-yhteyttä enää ole, tapetaan sekä dropbar että dbclient ja poistetaan tiedosto /tmp/remoteshell.txt. Näin ollaan jälleen alkupisteessä.

Skriptin kirjoittamisen ohessa suoritettujen testien yhteydessä törmättiin kuitenkin sellaiseen ongelmaan, että jos yhteys katkeaa, mutta se saadaan muodostettua muutamissa minuuteissa uudelleen, tunneli ei muodostukaan uudestaan, vaikka se näennäisesti onkin käynnissä, kun seurataan päätelaitteen prosesseja. Lisätutkimukset osoittivat, että palvelin, johon tunneli muodostetaan, ei huomaa tunnelin katkeamista heti vaan vasta jonkin ajan päästä. Tämä aika vaihteli noin 15 minuutista 20 minuuttiin, joten kyse on selkeästi jostain aikakatkaisumekanismista, joka on säädetty 15 minuuttiin ja ajastinta tarkkaillaan 5 minuutin aikavälein. Tämän esimerkin ssh.company.com koneena toimi testeissä Debian GNU/Linux, johon on asennettu OpenSSH-ohjelmisto ja Apache WWW-palvelin. OpenSSH:n asetustiedostoissa ei ollut mitään tähän viittaavaa, eikä lähdekoodia lähdetty sen tarkemmin tutkimaan, vaan tyydyttiin toteamaan, että mikäli päätelaitteelta katkeaa yhteys, on odotettava varmuuden vuoksi 30 minuuttia, ennen kuin uutta tunnelia lähdetään muodostamaan. Skriptiin lisättiin muuttuja conn_lost, jolla ilmaistaan, että jos yhteys katkeaa, ja dbclient oli ajossa, laitetaan muuttujan arvoksi 1. Muuttuja conn_lost alustetaan myös heti alussa arvoon 1, koska laite voi käynnistyä uudestaan käyttäjän tai sähkökatkon toimesta, jolloin ollaan samassa tilanteessa kuin yhteyden katkeamisen tapauksessa eli laite yrittäisi muuten muodostaa heti tunnelin, vaikka se ei onnistu, koska palvelin odottelee vielä edellisen tunnelin päässä. Kun seuraavan kerran yhteys palaa, ja huomataan yhteyden katkenneen viime kerralla, nukutaan 1800 sekuntia, jonka jälkeen conn_lost-muuttujan arvo palautetaan arvoon 0. Tämän korjauksen jälkeen tunneli on muodostunut jokaisella testauskerralla. start-dropbear.sh-skripti on luettavissa kokonaisuudessaan Internetissä [33].

Seuraavaksi tarvitaan skripti, jolla päivityspaketti voidaan asentaa päätelaitteen tiedostojärjestelmään. Tehdään tätä tarkoitusta varten check-updates.sh-skripti.

Skripti ajetaan myös järjestelmän käynnistysskriptistä, mutta päivitystoimenpide tehdään vain kertaluontoisesti, joten skriptiin ei tarvitse tehdä ikuista silmukkaa. Päätelaite on ajastettu käynnistymään uudestaan kerran vuorokaudessa, joten skripti ajetaan siten kerran vuorokaudessa, mikä on tarpeeksi tiheä aikaväli. Päätettiin, että jos hakemistoon `/root` ilmestyy tiedosto, siirretään se tiedostojärjestelmän juureen ja puretaan. Päivityspaketin tulee sisältää `tar`-ohjelmistolla arkistoitu hakemistorakenne, joka on vielä pakattu `gzip`-ohjelmistolla. Skriptiin on tulevaisuuden varalle rakennettu mahdollisuus tarkistaa myös MD5-tarkistussumma, jolla voidaan varmistua, että tiedosto on siirtynyt virheittä ja on täsmälleen sama, jonka paketin tekijä on tarkoittanutkin siirrettäväksi. TCP:n tehtäviin kuuluu siirtää tieto oikein, joten jätetään toistaiseksi MD5-tarkistussumma pois tämän työn tapauksesta. `check_updates.sh`-skripti on luettavissa kokonaisuudessaan Internetissä [34].

Edellisissä kappaleissa mainittiin, että skriptit käynnistetään järjestelmän käynnistysskriptistä, joten tiedosto `rcS` tarvitsee vain kaksi lisäriviä, jotka ovat

```
/rw/check-updates.sh
/rw/start-dropbear.sh &.
```

Skripti `check-updates.sh` suoritetaan luonnollisesti ensin, koska tarvittavat ohjelmat tulee asentaa ennen kuin niitä voi suorittaa. Skripti `start-dropbear.sh` käynnistetään taustalle lisäten `&`-merkki perään, jotta käynnistysskripti voi jatkaa viiveittä loppuun asti.

Koska päätelaite on sulautettu järjestelmä, siinä ei varsinaisesti tarvita minkäänlaista käyttäjähallintaa, mutta kun kirjaudutaan SSH:lla päätelaitteelle, siinä tulee olla luotuna jokin käyttäjä ja sillä salasana. Käyttäjän luominen onnistuu luomalla tiedosto `passwd`, joka sisällytetään päivityspakettiin. Käyttäjätunnuksen voi luoda `adduser`-ohjelmaa käyttäen komennolla

```
adduser root,
```

jonka jälkeen ohjelma kysyy luotavalle käyttäjälle salasanan. Järjestelmässä tulee olla vähintäänkin pääkäyttäjä, jolla on täydet oikeudet järjestelmään. Pääkäyttäjä Linux-järjestelmissä on yleisesti `root`. Komento tulee suorittaa jollain päätelaitteella, koska normaaleissa Linux-jakeluissa pääkäyttäjä on jo luotuna.

Nyt on luotu kaikki tarvittavat tiedostot, joten ne voidaan paketoida. Luodaan jollekin koneelle seuraavanlainen hakemistorakenne tarvittavine tiedostoineen:

```
/usr/sbin/dropbear
/usr/sbin/dbclient
/rw/start-dropbear.sh
/etc/passwd
/etc/dropbear/dropbear_rsa_host_key.
```

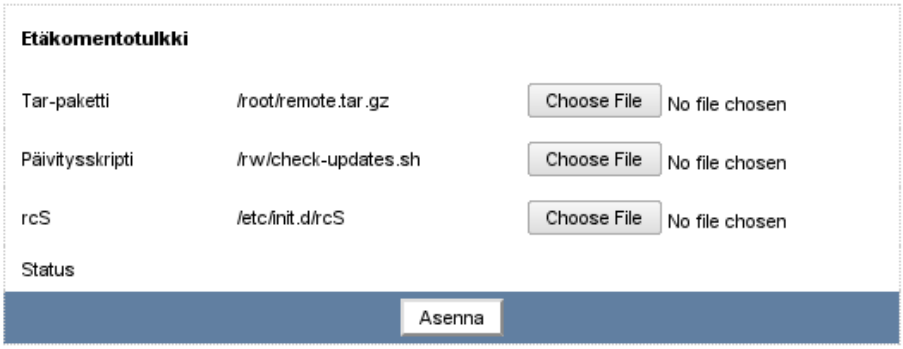
Pakettiin sisällytetään sekä SSH-palvelin että -asiakas, tunnelinmuodostusskripti, salasanatiedosto ja päätelaitteen salainen avain. Nämä paketoidaan luodun hakemistorakenteen juuressa komennolla

```
tar zcvf ../remote.tar.gz .,
```

joka tekee `remote.tar.gz` nimisen paketin yhtä hakemistoa alemmas ja sisällöksi laitetaan juurihakemiston sisältö, jota piste edustaa.

Vielä puuttuu käyttöliittymästä sivu, josta tarvittavat tiedostot voidaan syöttää lähetettäväksi päätelaitteelle. Sivulta pitää pystyä siis syöttämään tiedostot

`remote.tar.gz`, `check-updates.sh` ja `rcS`. Tehdään dOGMA-palvelinohjelmistoon JSP-sivu, jossa on kolme tiedostonsyöttökenttää. Kun tiedostot on otettu vastaan, siirretään ne Beaver-palveluun, josta ne lähetetään päätelaitteelle, mikäli yhteys on sillä hetkellä auki. Jos yhteys ei ole auki, tiedostot jäävät jonoon, kunnes päätelaite ottaa seuraavan kerran yhteyden.



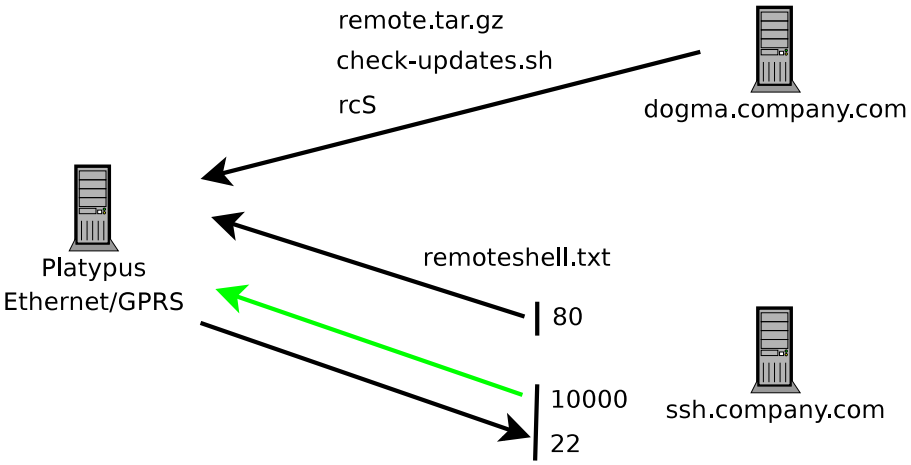
Kuva 13: SSH-ohjelmiston asennussivu käyttöliittymässä

Kuvassa 13 näkyy kuvankaappaus valmiista SSH-ohjelmiston asennussivusta. Tiedosto `rcS` korvaa olemassaolevan käynnistysskriptin. Tiedosto `check-updates.sh` siirretään hakemistoon `/rw`, jossa sijaitsee Oliotalon ohjelmistoa. Päivityspaketti `remote.tar.gz` siirretään hakemistoon `/root`, josta `check-updates.sh` havaitsee seuraavan käynnistyksen yhteydessä paketin ja asentaa sen.

Viimeisenä ratkaisua tulee luonnollisesti testata, jotta voidaan varmistua sen toimivuudesta. Tätä varten Oliotalon laboratorioon asennettiin yksi päätelaite, joka kytkettiin Ethernetillä lähiverkkoon. Päätelaitteeseen asennettiin myös SIM-kortti, jotta GPRS-yhteydenkin käyttäminen on mahdollista. Päätelaite konfiguroitiin ottamaan yhteys Oliotalon dOGMA-palvelinohjelmiston kehityspalvelimelle. Yksi Oliotalon palvelin konfiguroitiin SSH- ja WWW-palvelimeksi, jonne päätelaitteiden on tarkoitus muodostaa käänteinen tunneli. SSH-palvelimelle tehtiin käyttäjä, jonka kotihakemistoon asennettiin päätelaitteen julkinen salausavain aiemmin käytyjen ohjeiden perusteella. WWW-palvelimelle luotiin hakemisto, jonka nimi on sarjanumero, jollaisen laboratorioon asennettu päätelaite omaa. Hakemistoon luotiin `remoteshell.txt`, jonka sisällöksi laitettiin SSH-palvelimen osoite ja portiksi `10000`. WWW-palvelin kuuntelee yhteyksiä portissa 80 ja SSH-palvelin portissa 22.

Kuvassa 14 nähdään testausasetelma. SSH-ohjelmiston asennukseen tarvittavat tiedostot lähetetään palvelimelta `dogma.company.com`. Kun ohjelmisto on asentunut, päätelaite hakee `remoteshell.txt`-tiedoston koneen `ssh.company.com` WWW-palvelimelta, jonne päätelaite tässä tapauksessa luo myös käänteisen tunnelin, koska sama kone toimii SSH-palvelimena.

Taulukossa 6 nähdään testitapaukset ja niiden tulokset. Yhteyksien katkaisu tapahtui poistamalla päätelaitteelta käsin joko tiedosto `/tmp/eth-wanted` tai `/tmp/ppp--wanted`, jolloin yhteysskriptit lopettivat sen hetkisen yhteyden, ja mittausohjelmiston logiikka vaihtoi yhteystapaa. Tunnelin muodostuminen todettiin kirjautumalla



Kuva 14: Testausasetelma

Taulukko 6: Laboratoriotestien tulokset.

Toiminne	**Tulos**
SSH-ohjelmiston asennus.	Onnistui.
Tunnelin muodostus Ethernet-yhteydellä.	Onnistui.
Ethernet-yhteyden katkaisu.	Päätelaite vaihtoi GPRS-yhteyteen ja muodosti tunnelin uudelleen 30 minuutin kuluttua.
GPRS-yhteyden katkaisu.	Päätelaite vaihtoi Ethernet-yhteyteen ja muodosti tunnelin uudelleen 30 minuutin kuluttua.
Poistettiin remoteshell.txt WWW-palvelimelta ja käynnistettiin päätelaite uudelleen.	Tunneli ei muodostunut, mikä on oikea tulos.
Lisättiin remoteshell.txt WWW-palvelimelle.	Tunneli muodostui 10 minuutin aikana.
Linux-ytimen ja ajurimoduulien päivitys	Onnistui.

koneelta `ssh.company.com` päätelaitteelle komennolla

```
ssh -l root -p 10000 localhost,
```

jolloin salasanan antamisen jälkeen oli yhteys päätelaitteen komentoriville. Linux-ytimen ja ajurimoduulien päivitys tehtiin siten, että tarvittavat tiedostot ladattiin WWW-palvelimelta `wget`-ohjelmistolla, minkä jälkeen ne asennettiin, ja päätelaite käynnistettiin uudelleen, jolloin se käynnistyi uudella Linux-ytimellä onnistuneesti.

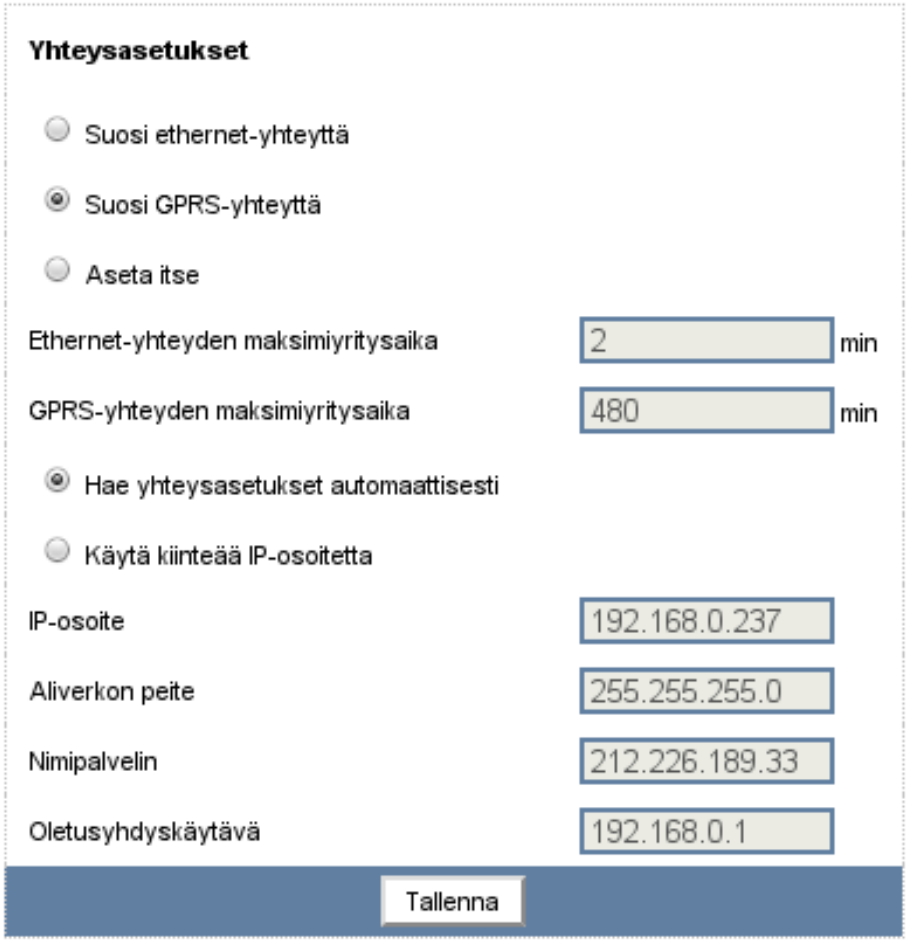
Komentorivin käyttäminen GPRS-yhteyden yli oli yllättävän sulavaa, vaikkakin hieman hitaampaa kuin Ethernet-yhteyttä käytettäessä. Taulukosta 6 voidaan havaita, että kaikki testit menivät läpi onnistuneesti, joten ratkaisua voidaan pitää onnistuneena. Projektin alkaessa oli varauduttu siihen, että mittausohjelmistoon tehtäisiin logiikka, joka rajoittaisi viestiliikennettä siksi aikaa, kun tunneli on käynnissä. Viestiliikenne on kuitenkin niin vähäistä, ettei se aiheuta ongelmaa tunnelin muodostuksessa tai sen käytettävyydessä. Sen sijaan mittausohjelmistoon joudutaan tekemään päivitys, koska tunnelin muodostus on mahdotonta, mikäli lähiverkon reunalla on palomuuri, josta on estetty liikennöinti muiden porttien kautta kuin on erikseen avattu M2M-liikenteelle. Tällaisia palomuurikonfiguraatioita on havaittu olevan kentällä, joten SSH-ohjelmisto on turha, ellei sillä voi muodostaa tunnelia palomuurin ohitse.

6.3 Palomuurin kiertäminen

Tämä luku käsittelee mittausohjelmisto dOGMA:n yhteystapalogiikan muutoksia siten, että lähiverkon palomuuri Ethernet-yhteyttä käytettäessä voidaan kiertää tarvittaessa. Palomuurin kiertäminen tapahtuu yksinkertaisesti siten, että käsketään päätelaite käyttämään pelkästään GPRS-verkkoa, jossa ei ole havaittu olevan palomuureja esteenä SSH-liikenteelle.

Vanha yhteystapalogiikkahan toimi siten, että mikäli sillä hetkellä käytössä oleva yhteys katkeaa, vaihdetaan toiseen. Tämä on siinä mielessä hieman rajoittunut, koska yhteys voi katketa satunnaisesti mutta olla jälleen käyttökelpoinen uuden yhteydenmuodostuksen jälkeen. Uusi ratkaisu tulee perustumaan siihen, että päätelaitteelle voi konfiguroida maksimiajan, jonka se yrittää muodostaa yhteyttä tietyllä yhteystavalla, minkä jälkeen vasta vaihdetaan toiseen yhteystapaan. Tämä logiikka ei pelkästään auta palomuurin kiertämisessä vaan parantaa muutenkin Oliotalon M2M-alustan käytettävyyttä ja konfiguroitavuutta. Samalla päätettiin tehdä myös Ethernet-asetuksista etäkonfiguroitavat. Tätä varten tehtiin jälleen JSP-sivu dOGMA-palvelinohjelmistoon.

Kuvassa 15 nähdään kuvankaappaus luodusta konfigurointisivusta. Yhteystapa voidaan valita kolmesta eri vaihtoehdosta, joita ovat suosi Ethernet-yhteyttä, suosi GPRS-yhteyttä tai aseta itse. Ensisijaisen yhteystavan uudelleenyritysaika on 480 minuuttia, ja toissijaisen yhteystavan vastaava aika on 2 minuuttia. Nämä arvot ovat peräisin asiakkaalta, joten niiden valintaan ei ole mitään tieteellistä perustelua. Pääasia on, että toinen arvoista on merkittävästi suurempi, jolloin selkeästi suositaan toista yhteystapaa. Aseta itse-vaihtoehdolla arvot voidaan syöttää käsin. Jos yhteystavan suosiminen olisi toteutettu siten, että käsketään päätelaitetta käyttämään vain toista yhteyttä, olisi mahdollista, että suosittuun yhteystapaan tulisi häiriö, jolloin päätelaite ei pystyisi enää ikinä muodostamaan yhteyttä palvelimeen, jolloin se tarkoittaisi sitä, että jonkun täytyisi käydä fyysisesti paikan päällä. Maksimiyritysaikaan perustuva ratkaisu toipuu lähettämään mittaustietoa, vaikka suosittu yhteystapa olisikin epäkunnossa. Ethernet-asetukset voidaan määritellä joko niin, että päätelaite hakee ne automaattisesti DHCP-palvelimelta, tai ne määritellään kiinteästi käsin. Näitä asetuksia varten luotiin myös Sparrow-protokollaan uusi



Kuva 15: Yhteysasetusten määrittäminen käyttöliittymässä

elementti. Kun JSP-sivulta painaa Tallenna-nappia, tehdään Sparrow-protokollan mukainen viesti, ja siirretään se Beaver-palveluun lähetettäväksi päätelaitteelle.

dOGMA-päätelaiteohjelmistoon toteutettiin configurator-sm-tilakoneeseen tarvittavan Sparrow-elementin vastaanottaminen ja tietojen tallennus asetustiedostoon. Carrier-sm-tilakoneeseen tehtiin logiikkamuutos uusien asetusten pohjalta. Uusi logiikka toimii siten, että asetusten saapuessa pudotetaan nykyinen yhteys alas, jonka jälkeen katsotaan, kummassa yhteystavassa on suurempi uudelleenyritysajan arvo. Suuremman arvon omaava yhteystapa laitetaan muodostumaan aina ensin. Jos arvot ovat yhtä suuret, ensisijainen yhteystapa on Ethernet. Näin yhteystapaa voidaan vaihdella vapaasti. Tätä voidaan käyttää hyväksi palomuurin kiertämiseen siten, että kun normaalisti asiakas haluaa käyttää aina Ethernet-yhteyttä, mikäli sellainen on tarjolla, mutta käänteisen tunnelin muodostamiseksi yhteystapa voidaan vaihtaa helposti GPRS-yhteydeksi. Kun huoltotyöt on tehty, vaihdetaan yhteystapa sulavasti takaisin Ethernet-yhteydeksi. Kiinteää IP-osoitetta voidaan mahdollisesti hyödyntää sillä tavoin, että jos paikallisen lähiverkon hallinnoija suostuu avaamaan yhdelle tietylle IP-osoitteelle mahdollisuuden SSH-liikennöintiin, niin päätelaitteet

voidaan vuorotellen konfiguroida käyttämään tätä tiettyä IP-osoitetta, jolloin huoltotyöt mahdollistuvat tunnelin kautta. Kun huoltotyöt on tehty, konfiguroidaan päätelaite vain käyttämään vanhoja asetuksia. Näin päätelaitteet voidaan ikään kuin ajaa halliin sisään, tehdä huoltotyöt, ajaa hallista ulos ja tehdä seuraavalle samat toimenpiteet. Näin menetellessä lähiverkon hallinnoijankaan ei tarvitse olla huolissaan tietoturvasta, koska normaalisti IP-osoitetta ei käyttäisi kukaan, joten aikaikkuna tietomurron tekemiseen on häviävän pieni.

Toteutuksen jälkeen tämänkin ratkaisun oikeellisuus varmistettiin muutamalla laboratoriotestillä, jotta voitiin varmistua sen laadusta ennen kuin kenttätestit on mahdollista aloittaa. Testausasetelma oli samanlainen kuin edellisessä testissä.

Taulukosta 7 nähdään testitapaukset ja niiden tulokset. Testitapaukset on pyritty rakentamaan siten, että ne testaavat kattavasti ratkaisun kaikki eri vaihtoehdot. Päätelaitteen toimintaa seurattiin dOGMA-mittausohjelmiston lokitiedostosta. GPRS-yhteyden aitoa katkeamista on vaikea simuloida laboratoriossa, mutta testeissä käytetty tekniikka simuloi tarpeeksi hyvin katkeamisen, vaikka yhteys katkaistaankin päätelaitteen itsensä toimesta, mutta katkaisu tehdään niin, että mittausohjelmisto luulee sen tapahtuneen ulkoisten vaikutusten seurauksena. Jotta saataisiin aito simulaatio, radiosignaali pitäisi pimentää täysin päätelaitteen antennilta. Tätä varten ei lähdetty rakentamaan eristyslaatikkoa, koska todettiin, että käytetty tekniikka simuloi tarpeeksi hyvin katkeamista. Ethernet-yhteys on helppo katkaista irroittamalla kaapeli päätelaitteesta. Tulosten perusteella ratkaisu näyttää toimivan niin kuin oli tarkoituskin, joten kaikki toteutetut ohjelmistot ovat valmiita kenttätestejä varten.

6.4 Kenttätestit

Tässä luvussa laitetaan kehitetyt ohjelmistokomponentit varsinaiseen tulikokeeseen, eli suoritetaan kenttätestaus. Testaus suoritetaan aidossa tuotantoympäristössä. Testi sisältää SSH-ohjelmiston asennuksen päätelaitteelle, yhteystavan vaihtamisen tarpeen vaatiessa GPRS-yhteydeksi, kirjautumisen päätelaitteelle käyttäen käänteistä tunnelia, Linux-ytimen päivityksen ajurimoduuleineen ja lopuksi yhteystavan vaihtaminen takaisin Ethernet-yhteydeksi niiden päätelaitteiden osalta, joilla se oli käytössä ennen päivitystä. Testi on onnistunut, jos useampi ongelmallinen laite saadaan päivitettyä. Testiin valittiin vaihteleva joukko laitteita, jotka ovat sijoitettuna eri puolilla maapalloa.

Taulukosta 8 nähdään kenttätestien tulokset. Laitteita oli sijoitettuna kahdeksaan eri maahan ja päätelaitteet oli asennettuna neljään erityyppiseen valvottavaan kohteeseen. Kolme kohteista on paikallaan pysyviä ja yksi liikkuva. Liikkuvassa tapauksessa, joka tarkoittaa trukkia, voidaan huomata, että yhteys katkesi useammin kuin paikallaan olevien tapauksessa. Tämä johtunee siitä, että trukit liikkuvat varastossa jatkuvasti ja voivat silloin tällöin ajaa sellaiseen paikkaan, jossa ei ole tarpeeksi kuuluvuutta, ja GPRS-yhteys katkeaa. Kuitenkin tunneli muodostui uudestaan, ja päivitystä voitiin täten yrittää uudestaan. Lopulta päivitys aina onnistui. Yksikään laite ei jäänyt testeissä päivittymättä kokonaan, vaan joissakin katkenneissa tapauksissa päivitys lopulta onnistui useamman yrityskerran jälkeen. Tietyissä koh-

Taulukko 7: Yhteyskonfigurointiratkaisun testitulokset.

Toiminne	**Tulos**
Asetettiin asetus suosi Ethernet-yhteyttä ja hae yhteysasetukset automaattisesti.	Päätelaite muodosti Ethernet-yhteyden käyttäen DHCP:n tarjoamia asetuksia.
Asetettiin asetus käytä kiinteää IP-osoitetta, ja konfiguroitiin lähiverkon asetukset käsin.	Päätelaite muodosti Ethernet-yhteyden käyttäen annettuja asetuksia.
Katkaistiin Ethernet-yhteys kahdeksi tunniksi, minkä jälkeen kytkettiin yhteys takaisin.	Ethernet-yhteys katkesi, ja päätelaite yritti muodostaa kahden tunnin ajan Ethernet-yhteyttä. Kun Ethernet kytkettiin takaisin, yhteys muodostui onnistuneesti.
Asetettiin asetus suosi GPRS-yhteyttä.	Päätelaite muodosti GPRS-yhteyden.
Otettiin SIM-kortti pois päätelaitteesta kahdeksi tunniksi, ja poistettiin tiedosto /tmp/ppp-wanted.	GPRS-yhteys katkesi, ja päätelaite yritti muodostaa GPRS-yhteyttä kaksi tuntia mutta ei onnistunut.
Laitettiin SIM-kortti takaisin.	Päätelaite muodosti GPRS-yhteyden.
Asetettiin asetus aseta itse, annettiin arvot Ethernet 20 minuuttia ja GPRS 10 minuuttia.	Päätelaite muodosti Ethernet-yhteyden.
Katkaistiin Ethernet-yhteys.	Päätelaite yritti muodostaa Ethernet-yhteyttä 20 minuuttia, minkä jälkeen muodosti GPRS-yhteyden.
Kytkettiin Ethernet takaisin, otettiin SIM-kortti pois ja poistettiin tiedosto /tmp/ppp-wanted.	Päätelaite yritti muodostaa GPRS-yhteyttä 10 minuuttia, minkä jälkeen muodosti Ethernet-yhteyden.
Asetettiin arvoiksi Ethernet 10 minuuttia ja GPRS 10 minuuttia.	Päätelaite muodosti Ethernet-yhteyden.

teissa kuten Australiassa yhteystavaksi on tietoisesti valittu Ethernet, koska asennusvaiheessa on jo havaittu, että GPRS-verkon kuuluvuus ei ole riittävä sillä alueella, joten tästä syystä myös päivitystestissä päivitys ei onnistunut kummassakaan tapauksessa ensimmäisellä kerralla. Päivityksen hitain vaihe on siirtää päätelaitteelle Linux-ydin, joka on kooltaan huomattavan iso GPRS-verkon siirtokapasiteettiin nähden. Linux-ydin on kooltaan noin 2 Megatavua, joten sen siirtäminen huonolla yhteydellä (1 kt/s) kestää yli 30 minuuttia. Tässä ajassa trukki voi siirtyä monesti alueelle, jossa radiosignaali on heikko. Tällaiseen tilanteeseen ei voi etänä vaikuttaa,

Taulukko 8: Kenttätestauksen tulokset.

Sijainti	**Kohde**	**Määrä**	**Tulos**
Suomi	Tuulimylly	2	Onnistui
Suomi	Ruoka-automaatti	2	Onnistui
Suomi	Torninosturi	2	Onnistui
Saksa	Tuulimylly	10	Onnistui (useampi yritys 3:ssa)
Australia	Tuulimylly	2	Onnistui (useampi yritys 2:ssa)
Iso-Britannia	Tuulimylly	2	Onnistui
Espanja	Tuulimylly	2	Onnistui (useampi yritys 1:ssä)
Ranska	Tuulimylly	2	Onnistui
Yhdysvallat	Tuulimylly	1	Onnistui
Ruotsi	Trukki	10	Onnistui (useampi yritys 5:ssä)
Yhteensä		**35**	

vaan on vain elettävä sen kanssa, että yhteys välillä katkeaa. Yhteenvetona voidaan todeta, että kaikki 35 laitetta saatiin päivitettyä, joten onnistumisprosentti on sata. Tulos voidaan määritellä onnistuneeksi.

6.5 Yhteenveto

Luvun alussa määriteltiin, mitä toimenpiteitä käyttäjän tulee tehdä, että huoltoyhteys voidaan muodostaa. Tästä lähtökohdasta lähdettiin rakentamaan toteutusta. Ennen toteutuksen luontia käytiin pikaisesti läpi päätelaitteen tiedostojärjestelmän rakennetta ja järjestelmästä jo valmiiksi löytyviä yhteydenluontimekanismeja. Toteutus aloitettiin tarvittavien skriptien luonnilla. Kun skriptit oli kirjoitettu ja testattu toimiviksi, paketoitiin skriptit yhdessä SSH-ohjelmiston kanssa yhdeksi päivityspaketiksi. Vastaavasti palvelinohjelmistoon toteutettiin mekanismi, jolla päivityspaketti voitiin siirtää päätelaitteelle. Tälle toteutukselle suoritettiin Oliotalon laboratoriossa testaus, joka onnistui.

Luotu ratkaisu ei vielä toiminut tapauksissa, joissa organisaation verkossa on palomuuri, joka ei salli SSH-liikennettä. Tämän rajoituksen kiertämiseksi luotiin mekanismi, jolla päätelaite saadaan vaihtamaan yhteystapaa. Palvelinohjelmistoon kirjoitettiin konfigurointisivu, josta päätelaitteen yhteysasetuksia voitiin vaihtaa. Vastaavien asetusten vastaanotto ja uusi yhteystapalogiikka toteutettiin päätelaiteohjelmistoon. Tätäkin mekanismia testattiin laboratoriossa. Kun ratkaisu oli testattu toimivaksi, siirryttiin kenttätestaukseen. Kenttätestauksessa luotiin huoltoyhteys ympäri maapalloa sijaitseviin päätelaitteisiin, joihin päivitettiin Linux-ydin ajurimoduuleineen. Myös kenttätestaus onnistui, vaikka päätelaitteet eivät päivittyneetkään ensimmäisellä kerralla, mutta onnistumisprosentti oli lopulta sata.

7 Yhteenveto

Tämä työ aloitettiin tutustumalla kunnonvalvontaan yleisesti. Kunnonvalvonnasta opittiin tärkeimpänä se, että kyse on vauriontunnistamisesta. Tähän oli monta eri lähestymistapaa, mutta kaikille yhteistä oli pystyä tunnistamaan häiriö jonkin laitteen tai rakenteen toiminnassa ja ennakoida mahdollinen rikkoutuminen. Kunnonvalvonnalla todettiin olevan kaksi suurta hyötyä: raha ja turvallisuus. Kunnonvalvonnalla voidaan havaita ongelmat ajoissa ja korjata ne, jolloin tuotanto pysyy käynnissä katkottomasti, eikä suurempia taukoja tule laitteen korjauksen ajaksi. Toisaalta taas organisaatiot haluavat panostaa henkilöstön turvallisuuteen, koska kun tiedetään laitteiden olevan kunnossa, niitä on turvallista käyttää. Laiterikko avaruussukkulassa voi olla hyvinkin kohtalokasta matkustajille, joten riskejä ei haluta ottaa turhan takia, jos ne voidaan välttää. Samoin sillan romahtaminen voi olla kohtalokasta, jos sillalla on romahdushetkellä ihmisiä. Tällaiset onnettomuudet halutaan välttää kunnonvalvonnan avulla. Kunnonvalvonnan hyödyistä siirryttiin tarkastelemaan, miten kunnonvalvonta on toteutettu aiemmin, ja millaisia ratkaisuja nykyisin on tarjolla sen toteuttamiseksi. Tarkasteluissa havaittiin, että aiemmin kunnonvalvonta on perustunut akustisiin havaintoihin, kun rakennetta on naputeltu eri kohdista. Nykyisin elektroniikka on kehittynyt niin pitkälle, että havaintojen tekemiseen on tehty paljon erilaisia antureita, joilla voidaan havaita erilaisia fysiikan suureita. Näitä suureita voidaan lukea antureilta tietokoneella. Kehitys on myös mahdollistanut tietokoneiden kutistamisen niin pieneksi, että sen voi kiinnittää helposti valvottavan laitteen kylkeen. Tietoliikennetekniikan kehitys on puolestaan mahdollistanut osittain kaapeloinnin poistamisen valvottavasta laitteesta tai rakenteesta, jolloin liikkuvien ajoneuvojen valvontakin on mahdollista nykyään. Näistä havainnoista päädyttiin esittelemään erilaisia teknologioita, joita termi M2M verhoaa sisäänsä. Teknologioiden esittelyn jälkeen luotiin katsanto M2M:n sovelluksiin ja huomattiin, että M2M soveltuu monipuolisesti erilaisiin valvontatehtäviin, eikä rajoitu ainoastaan kunnonvalvontaan. Kun M2M oli tehty tutummaksi, syvennyttiin Oliotalon olemassa olevaan M2M-alustaan, jossa esiteltiin sekä käytössä olevaa laitteistoa että ohjelmistoa päätelaitteen ja palvelimen osalta. Tämä antoi konkreettisemman esityksen tiedon kulusta, joka alkoi päätelaitteella havaituista sähköisistä signaaleista ja päätyi palvelimelle vietäväksi tietokantaan, josta tieto voitiin hakea loppuen lopuksi asiakkaan selaimelle esitettynä hyödyllisessä muodossa.

Tässä vaiheessa huomattiin, että Oliotalonkaan nykyinen alusta ei ole täydellinen, vaan kehitystä tapahtuu jatkuvasti, jolloin myös jo kentälle toimitetut päätelaitteet vaativat päivitystä. Vaikka Oliotalolla oli jo olemassa ratkaisu mittausohjelmiston päivittämiseksi, huomattiin ettei se riitä, jos alla olevassa käyttöjärjestelmässä havaitaan ohjelmointivirhe. Päivitystä ei haluttu lähteä tekemään ympäri maapalloa paikan päälle, vaan järkevää oli pystyä tekemään päivitys keskitetysti yhdestä paikasta. Tästä johtuen Oliotalo halusi tutkia mahdollisuutta muodostaa päätelaitteille huoltoyhteys jollakin keinolla. Oliotalon aiempi kokemus asiakkaiden tietoverkoista osoitti, että huoltoyhteyden luominen ei olisi mahdollista toteuttaa suoraviivaisesti, vaan tarvittiin jokin älykäs toteutustapa. Tästä johtuen tutustuttiin aluksi yleisesti tietoverkkojen ja tietoliikenteen toimintaan ja yhteyksien luon-

tiin, minkä jälkeen katselmoitiin asiakkaiden tietoverkoissa häiritseviä elementtejä. Kun häiritsevät elementit olivat tiedossa, arvioitiin SSH:n soveltuvuutta työn osaratkaisuksi. Opittiin, että se soveltuu mainiosti tarkoitukseen, jossa päätelaitteelle on tarkoitus luoda etäkomentokehoteyhteys. Tämä onnistuu luomalla käänteinen tunneli, jota pitkin joltain keskitetyltä palvelimelta on mahdollista kirjautua eri päätelaitteille. Tämän pohjalta lähdettiin kehittelemään varsinaista ratkaisua. Tehtävänä oli toteuttaa muutama skripti, joilla tunneli voitiin muodostaa automaattisesti siten, että se ottaa huomioon myös sen, että yhteys voi välillä katketa. SSH-ohjelmiston asentamiseksi luotiin palvelinohjelmistoon sivusto ja mekanismit, joilla päivityspaketti voitiin siirtää päätelaitteelle. Ratkaisua testattiin ensin laboratoriossa, minkä jälkeen se todettiin toimivaksi. Silti edessä oli vielä yksi haaste, joka oli asiakkaiden lähiverkoissa toimivat palomuurit, jotka on konfiguroitu päästämään vain tietty liikenne läpi, ja siihen ei kuulunut SSH-liikenne. Palomuurien kiertämiseksi toteutettiin päätelaiteohjelmistoon logiikkamuutos, jolla voitiin pakottaa päätelaite vaihtamaan yhteystapaa Ethernetistä GPRS:ksi, koska GPRS-verkossa ei ole havaittu palomuureja olevan välissä. GPRS-verkko on ainoastaan osoitteenmuunnoksen takana, johon jo toteutettu käänteisen tunnelin ratkaisu oli toimivaksi havaittu. Asetusten syöttämistä varten palvelinohjelmistoon luotiin uusi konfigurointisivu, josta voitiin vaihtaa yhteystapaa ja lisäksi asettaa käsin Ethernet-asetukset. Asetusten siirtäminen palvelimelta päätelaitteelle vaati myös uuden elementin Oliotalon käyttämään Sparrow-protokollaan. Luotuja uudistuksia testattiin jälleen ensin laboratoriossa. Kaikki huoltoyhteyden vaatimat ohjelmistot todettiin toimiviksi laboratoriossa, joten oli aika suorittaa kenttätestaus, jossa ohjelmistojen todellinen suorituskyky mitattiin. Kenttätestaus suoritettiin aidossa tuotantoympäristössä. Kenttätestauksessa päivitettiin Linux-ydin ja ajurimoduulit yhteensä 35 päätelaitteeseen. Laitteet sijaitsivat kahdeksassa eri maassa, ja päätelaite oli asennettuna neljään eri tyyppiseen kohteeseen, joista kolme oli paikallaan pysyviä ja yksi liikkuva. Kaikki 35 päätelaitetta saatiin päivitettyä, joskaan kaikki eivät päivittyneet ensimmäisellä yrittämällä, koska osasta päätelaitteita katkesi yhteys kesken päivityksen. Juuri tätä varten ratkaisu rakennettiin siten, että se kestää yhteyskatkot. Loppuen lopuksi onnistumisprosentti oli sata.

Projektia voidaan pitää onnistuneena, koska alkuperäinen tavoite oli tutkia mahdollisuutta muodostaa huoltoyhteys hajasijoitetuille päätelaitteille keskitetysti, ja tässä tavoitteessa onnistuttiin. Nyt Oliotalolla on ratkaisu, jolla voidaan kirjautua päätelaitteille etänä ja tehdä mikä tahansa ohjelmistopäivitys käymättä fyysisesti paikan päällä. Kehitetty ratkaisu on vielä raaka prototyyppi, joten sen ympärille on mahdollista kehittää vielä paljon automatisoidumpi ratkaisu. Päätelaitteen SSH-ohjelmisto voisi esimerkiksi olla osana jo tehdasasenteista juuritiedostojärjestelmää, jolloin sitä ei tarvitsisi erikseen siirtää palvelimelta päätelaitteelle. SSH- ja WWW-palvelinkonetta varten voisi kehittää hallintaohjelmiston, jolla hallitaan etäyhteyksien aukaisemista ja sulkemista. Tämä tarkoittaisi käytännössä sarjanumeroon perustuvan hakemistorakenteen hallinnoimista ja hakemistoista löytyvien tekstitiedostojen hallinnoimista. Hallintaohjelmisto voisi tarkkailla, milloin käänteinen tunneli aktivoituu, jolloin olisi mahdollista asettaa vaikkapa sähköpostihälytys, että nyt on mahdollista kirjautua päätelaitteelle tekemään huoltotyöt. Mahdollisuuksia on mo-

nia. Oliotalolla ei kuitenkaan ole tällä hetkellä suunnitelmissa tämän kaltaisen ohjelmiston kehittämistä. Sen sijaan projektista saatiin Linux-ytimen etäpäivityksen onnistumisen seurauksena sen verran positiivinen signaali, että tästä rohkaistuneena Oliotalo on aloittanut kehittämään Linux-ytimen etäpäivityksen massatoimintoa. Vaikkakin tässä työssä kehitetty etähuoltoyhteys on toimiva, se vaatii huomattavan paljon huomiota ja valvontaa Oliotalon henkilökunnalta seurata käänteisen tunnelin muodostumista ja päivitysten etenemistä. Tämä ei muodostu ongelmaksi muutaman päätelaitteen päivityksessä, mutta silloin kun pitää päivittää esimerkiksi tuhat laitetta, se alkaa jo muodostua melkoiseksi kustannukseksi, koska tehtävät toimenpiteet sitovat turhaan työvoimaa projekteista, joista voidaan laskuttaa. Ohjelmointivirheestä johtuva päivitys on useimmiten takuunalaista työtä, josta ei saa korvausta. Tästä syystä on kustannustehokkaampaa ohjelmoida palvelimelle logiikka, joka siirtää Linux-ytimen ja ajurimoduulit päätelaitteelle käyttäen TCP-yhteyttä, joka on muutenkin auki valvontatiedon lähettämistä varten. Linux-ytimen ja ajurimoduulien päivitys on mahdollista toteuttaa skripteillä päätelaitteessa. Käyttäjää varten tehdään palvelimelle sivu, josta esimerkiksi ruksataan vain päivitettävät päätelaitteet, ja palvelin tekee kaiken muun työn. Näin henkilöstöresurssit vapautuvat muuhun käyttöön.

Viitteet

[1] Farrar, Charles R. ja Worden, Keith. An introduction to structural health monitoring. Phil. Trans. R. Soc. A (2007) 365, s. 303–315. Verkkodokumentti. Julkaistu 12.12.2006. Viitattu 16.12.2009. Saatavissa: http://institute.lanl.gov/ei/shm/pubs/PTRS%20Intro%2006.pdf. DOI: 10.1098/rsta.2006.1928.

[2] United States Department of Defense. MIL-STD-271F(SH), Military Standard, Requirements for Nondestructive Testing Methods. Verkkodokumentti. Julkaistu 27.6.1986. Viitattu 17.12.2009. Saatavissa: http://www.everyspec.com/MIL-STD/MIL-STD+(0100+-+0299)/download.php?spec=MIL-ST

[3] A Summary of RFID Standards. RFID Journal, Inc, 2005. Verkkodokumentti. Julkaistu 2005. Viitattu 23.12.2009. Saatavissa: http://www.rfidjournal.com/article/pdf/1335/1/1/rfidjournal-article1335.PDF.

[4] IEEE Std 802.11-2007. Part 11: Wireless LAN Medium Access Control(MAC) and Physical Layer (PHY) Specifications. IEEE 3 Park Avenue New York, NY 10016-5997, USA. IEEE Computer Society, 2007. 1232 s. Myös sähköisessä muodossa saatavissa: http://standards.ieee.org/getieee802/download/802.11-2007.pdf.

[5] IEEE Std 802.15.4-2006. Part 15.4: Wireless Medium Access Control (MAC) and Physical Layer (PHY) Specifications for Low-Rate Wireless Personal Area Networks (WPANs). IEEE 3 Park Avenue New York, NY 10016-5997, USA. IEEE Computer Society, 2006. 323 s. Myös sähköisessä muodossa saatavissa: http://standards.ieee.org/getieee802/download/802.15.4-2006.pdf.

[6] Römer, Kay ja Friedemann, Mattern. The Design Space of Wireless Sensor Networks. *IEEE Wireless Communications,* 2004, vol. 11, nro 6, s. 54–61.

[7] Corvallis Microtechnology, Inc. Introduction to the Global Positioning System for GIS and TRAVERSE. Verkkosivu. Julkaistu kesäkuu, 1996. Viitattu 17.1.2010. Saatavissa: http://www.cmtinc.com/gpsbook/.

[8] IEEE Std 802.3-2008. Part 3: Carrier sense multiple access with Collision Detection (CSMA/CD) Access Method and Physical Layer Specifications (Section 1). IEEE 3 Park Avenue New York, NY 10016-5997, USA. IEEE Computer Society, 2008. 671 s. Myös sähköisessä muodossa saatavissa: http://standards.ieee.org/getieee802/download/802.3-2008_section1.pdf.

[9] IEEE Std 802.3-2008. Part 3: Carrier sense multiple access with Collision Detection (CSMA/CD) Access Method and Physical Layer Specifications (Section 2). IEEE 3 Park Avenue New York, NY 10016-5997, USA. IEEE Computer Society, 2008. 790 s. Myös sähköisessä muodossa saatavissa: http://standards.ieee.org/getieee802/download/802.3-2008_section2.pdf.

[10] IEEE Std 802.3-2008. Part 3: Carrier sense multiple access with Collision Detection (CSMA/CD) Access Method and Physical Layer Specifications (Section 3). IEEE 3 Park Avenue New York, NY 10016-5997, USA. IEEE Computer Society, 2008. 315 s. Myös sähköisessä muodossa saatavissa: http://standards.ieee.org/getieee802/download/802.3-2008_section3.pdf.

[11] IEEE Std 802.3-2008. Part 3: Carrier sense multiple access with Collision Detection (CSMA/CD) Access Method and Physical Layer Specifications (Section 4). IEEE 3 Park Avenue New York, NY 10016-5997, USA. IEEE Computer Society, 2008. 586 s. Myös sähköisessä muodossa saatavissa: http://standards.ieee.org/getieee802/download/802.3-2008_section4.pdf.

[12] IEEE Std 802.3-2008. Part 3: Carrier sense multiple access with Collision Detection (CSMA/CD) Access Method and Physical Layer Specifications (Section 5). IEEE 3 Park Avenue New York, NY 10016-5997, USA. IEEE Computer Society, 2008. 615 s. Myös sähköisessä muodossa saatavissa: http://standards.ieee.org/getieee802/download/802.3-2008_section5.pdf.

[13] http://m2m.orangeom.com/ verkkosivusto. M2M for oil and gas. Verkkosivu. Julkaistu 20.7.2007. Viitattu 17.1.2010. Saatavissa: http://m2m.orangeom.com/m2m-for-oil-and-gas/.

[14] Inglesby, Tom. M2M is LEEDing the Way. Verkkolehti. Julkaistu lokakuu, 2008. Viitattu 17.1.2010. Saatavissa: http://www.m2mmag.com/issue_archives/story.aspx?ID=7214.

[15] Comtech M2M. Vending telemetry offers remote monitoring via the Internet. Verkkosivu. Julkaistu 2008. Viitattu 17.1.2010. Saatavissa: http://www.comtechm2m.com/m2m-telemetry-solutions/vending-telemetry-solution.ht

[16] http://m2m.orangeom.com/ verkkosivusto. Remote Patient Monitoring (M2M in Healthcare). Verkkosivu. Julkaistu 8.5.2007. Viitattu 17.1.2010. Saatavissa: http://m2m.orangeom.com/remote-patient-monitoring-m2m-in-healthcare/.

[17] Häggblom, Kaius, Wirzenius, Lars ja Oksanen, Kenneth. Hedgehog. Verkkosivu. Viitattu 12.4.2010. Saatavissa: http://hedgehog.oliotalo.fi/.

[18] http://www.processor-comparison.com/ verkkosivusto. CPU Comparison Chart. Verkkosivu. Viitattu 2.2.2010. Saatavissa: http://www.processor-comparison.com/power.html.

[19] The Modbus Organization. Modbus Application Protocol Specification V1.1b. Verkkodokumentti. Julkaistu 28.12.2006. Viitattu 2.2.2010. Saatavissa: http://www.modbus.org/docs/Modbus_Application_Protocol_V1_1b.pdf.

[20] The Modbus Organization. Modbus over Serial Line Specification and Implementation Guide V1.02. Verkkodokumentti. Julkaistu 20.12.2006. Viitattu 2.2.2010. Saatavissa: http://www.modbus.org/docs/Modbus_over_serial_line_V1_02.pdf.

[21] Lindh, Tuomo. Kenttäväylät. Luentomateriaalia, Lappeenrannan teknillinen yliopisto. Viitattu 2.2.2010. Saatavissa: http://www.ee.lut.fi/courses/Sa2731500/CAN.pdf.

[22] Sähköturvallisuuden edistämiskeskuksen verkkosivusto. IP-numeroiden merkitys. Verkkosivu. Viitattu 28.1.2010. Saatavissa: http://www.sahkoturva.info/sahkon_kaytto_kotona/sahkolaitteiden_ip_luokitus/fi_F

[23] Lunz, Jason. Linux Kernel Mailing List. Verkkodokumentti. Julkaistu 30.8.2007. Viitattu 2.2.2010. Saatavissa: http://lkml.indiana.edu/hypermail/linux/kernel/0708.3/2084.html.

[24] Information Sciences Institute University of Southern California. Transmission Control Protocol Functional Specification. Internet Engineering Task Force, RFC 793, syyskuu, 1981.

[25] Braden, R. Requirements for Internet Hosts – Communication Layers. Internet Engineering Task Force, RFC 1122, lokakuu, 1989.

[26] Ingham, Kenneth ja Forrest, Stephanie. A History and Survey of Network Firewalls. Verkkodokumentti. Julkaistu 2002. Viitattu 11.2.2010. Saatavissa: http://www.cs.unm.edu/~treport/tr/02-12/firewall.pdf.

[27] Fielding, R., Gettys, J., Mogul, J., Frystyk, H., Berners-Lee, T. Hypertext Transfer Protocol – HTTP/1.1. Internet Engineering Task Force, RFC 2068, tammikuu, 1997.

[28] Ebox platform. HTTP Proxy Service. Verkkosivusto. Julkaistu 13.2.2010. Viitattu 17.3.2010. Saatavissa: http://doc.ebox-platform.com/en/1.2/proxy.html.

[29] Rekhter, Y., Moskowitz, B., Karrenberg, D., de Groot, G. J. ja Lear, E. Address Allocation for Private Internets. Internet Engineering Task Force, RFC 1918, helmikuu, 1996.

[30] Hain, T. Architectural Implications of NAT. Internet Engineering Task Force, RFC 2993, marraskuu, 2000.

[31] gregkelly. Publicly Addressed DMZ Infrastructure. Verkkodokumentti. Julkaistu 25.2.2009. Viitattu 24.2.2010. Saatavissa: http://wiki.oracle.com/page/Chapter+3+-+Publicly+Addressed+DMZ+Infrastructure.

[32] Johnston, Matt. Dropbear SSH server and client. Verkkosivusto. Julkaistu 6.5.2003. Päivitetty 12.11.2008. Viitattu 3.3.2010. Saatavissa: http://matt.ucc.asn.au/dropbear/dropbear.html.

[33] Lahti, Pasi. SSH-tunnelin hallintaskripti. Verkkosivusto. Julkaistu 10.3.2010. Päivitetty 12.4.2010. Viitattu 12.4.2010. Saatavissa: http://tux.oliotalo.fi/remote/scripts/start-dropbear.sh.txt.

[34] Dolgov, Ivan ja Lahti, Pasi. Päivityspaketin asennusskripti. Verkkosivusto. Julkaistu 10.3.2010. Päivitetty 10.3.2010. Viitattu 10.3.2010. Saatavissa: http://tux.oliotalo.fi/remote/scripts/check-updates.sh.txt.