TAMPEREEN TEKNILLINEN YLIOPISTO
Sähkötekniikan koulutusohjelma

MARKKU KOPONEN

TELCONT SECURE -HÄLYTYKSENSIIRTOLAITTEEN OHJELMISTO KETTERÄNÄ PROJEKTINA

Diplomityö

Tarkastaja: Hannu-Matti Järvinen
Tarkastaja ja aihe hyväksytty
tieto- ja sähkötekniikan tiedekunta-
neuvoston kokouksessa 03.03.2010

TIIVISTELMÄ

TAMPEREEN TEKNILLINEN YLIOPISTO
Sähkötekniikan koulutusohjelma
MARKKU KOPONEN: Telcont Secure -hälytyksensiirtolaitteen ohjelmisto ketteränä projektina
Diplomityö, 67 sivua, 3 liitesivua
Maaliskuu 2010
Pääaine: Ohjelmistotekniikka
Tarkastajat: Hannu-Matti Järvinen
Avainsanat: Hälytyksensiirto, sulautetut järjestelmät

Telcont Oy perustettiin julkisessa IP-verkossa tapahtuvaa, viranomaishyväksyttyä ilmoituksensiirtoa varten. Tässä työssä kuvataan siirtoverkon asiakaslaitetta, *ATM0602*-moduulia, jonka ohjelmiston kirjoittaja suunnitteli ja toteutti. Olemassa oleva ilmoituksensiirtotekniikka perustuu pelkästään siihen tarkoitukseen varattuihin kaapeliyhteyksiin, kun taas teknologinen kehityspaine Suomessa on ollut jo pitkään kohti pienempää määrää kaapeleita.

Olemassa oleva tekniikka on suunniteltu 70-luvulla ja ilmoituksensiirtolaitteiden on toimittava sen asettamien rajoitusten ehdoilla. Esimerkiksi osoitteistus koko Suomen kattavassa verkossa on vain 32-bittinen. Vaikka yhteensopivuuden takaamiseksi olikin hyväksyttävä nämä rajoitukset, uudistettiin kuitenkin ilmoituksensiirtoa siirtymällä piirikytkentäisestä ISDN-tekniikasta pakettikytkentäisiin verkkoihin, kuten ADSL-päätelaitteella käytettävään internet-yhteyteen. Tästä koituu asiakkaalle kustannussäästöjä. Pakettikytkentäisyyteen liittyy omat haasteensa, kuten tietoturvaongelmat.

ATM0602 toteutettiin irrottavana reaaliaikajärjestelmänä. Se takaa ohjelman suorituksen ennustettavuuden, mutta vaatii myös ohjelmoijalta tarkkuutta, sillä ohjelmavirheiden seuraukset voivat olla irrottavassa reaaliaikajärjestelmässä vakavia. Näihin ongelmiin on olemassa yleisesti tunnettuja ratkaisuja, kuten vahtikoira-ajastimen käyttäminen. Ohjelmistossa oli myös varauduttava useisiin, alan kirjallisuudesta löytyviin tunkeutumismenetelmiin.

Kehitysympäristönä toimi DynamicC, joka on työssä käytettyä RCM3700-moduulia valmistavan yrityksen tarjoama ohjelmisto. Kirjoittajan vastuulla oli toteuttaa ATM0602:n tukema, oma yhteyskäytäntö, joka oli työn alussa jo osittain määritelty. Tähän kuului sellaisia asioita kuin salauksen toteuttaminen, toistosanomat sekä varayhteydelle siirtyminen. Myös laitteen käyttöliittymän suunnitteli ja toteutti kirjoittaja. Työ toteutti sille asetetut odotukset ja ATM0602 on parhaillaan laajasti käytettynä eri puolilla Suomea.

ABSTRACT

TAMPERE UNIVERSITY OF TECHNOLOGY
Master's Degree Programme in Electrical Engineering Technology
MARKKU KOPONEN: Firmware of Telcont Secure Alarm Transmitter Device as an Agile Project
Master of Science Thesis, 67 pages, 3 Appendix pages
March 2010
Major: Software Systems
Examiner: Hannu-Matti Järvinen
Keywords: Alarm transmission, embedded systems

The company Telcont Oy was founded for developing a certified system for alarm transmission over public IP network. This thesis describes the development and implementation of a client device for the system, *ATM0602* module. Currently existing alarm transmission technology is based on dedicated cable connections between two points, whereas technological pressure in Finland has been towards fewer cables.

Existing technology was designed in the '70s, which places some restrictions on all alarm transmission devices, such as the 32-bit address space of the nationwide network. This was unavoidable in order to guarantee compatibility, but there was room for innovation in moving from circuit switched data to packet switched data, using an Internet connection through, for example, an ADSL modem. The result is savings for the customer. There are, however, certain challenges that come along with packet switching, such as security issues.

ATM0602 was implemented as a pre-emptive real-time system. This guarantees the predictability of program execution, but requires great care from the programmer. Consequences of errors in design or programming can be critical. There are some widely known solutions for mitigating such errors, such as a watchdog timer. There are also many hacking methods to be found in relevant literature that had to be prevented.

Development was done using DynamicC, a software offered by the manufacturer of RCM3700 module. RCM3700 was the hardware platform for ATM0602. Writer's responsibilities included implementing Telcont's own alarm transmission protocol, supported by the device. This involved such things as encryption, message retries and backup connection. The graphical user interface of the device was also designed and implemented by the writer. The work fulfilled its expectations and ATM0602 is currently widely used in Finland.

ALKUSANAT

Tämä on Markku Koposen diplomityö, joka käsittelee Telcont Oy:n valmistaman ilmoituksensiirtolaitteen, *ATM0602*-moduulin, suunnittelua ja toteutusta. Projekti alkoi vuoden 2006 kesällä, jolloin kirjoittaja oli kesätöissä Telcont Oy:ssa. Suurin osa ohjelmistokehityksestä tapahtui tuolloin. Tämän jälkeen kehitystyö tapahtui tuntiperustaisesti laskuttaen opiskelun ohessa, sitä mukaa kuin uusia ominaisuuksia tarvittiin, tai asiakkailta kantautui uutisia ongelmista.

Esitän kiitokset Telcont Oy:n perustajalle ja toimitusjohtajalle, Jussi Hakuliselle. Ilman hänen kymmenien vuosien kokemustaan ilmoituksensiirtojärjestelmistä ei projekti olisi ollut mahdollinen. Samaten kiitän Telcont Oy:n ohjelmistopäällikköä, Toni Röyhyä, jonka kanssa tiivistä yhteistyötä tekemällä saatiin aikaan toimiva ja innovatiivinen järjestelmä. Toni toimi projektin aikana ATS:n, eli ATM0602-moduuleja hallinnoivan palvelinlaitteen suunnittelijana ja ohjelmoijana.

Kiitokset myös Telcont Oy:n entiselle työntekijälle, Teijo Eilolalle, joka toteutti laitteen käyttämät ohjelmakirjastot GSM-signalointia, GPRS-yhteyttä sekä TCP/IP:llä liikennöintiä varten. Lisäksi mainittakoon, että Telcont Oy kustansi projektin aikana kirjoittajalle kolme Teleware Oy:n järjestämää kurssia.

SISALLYS

LYHENTEET JA TERMIT

ARP Address Resolution Protocol; protokolla laitteen MAC-osoitteen selvittämiseksi sen IP-osoitteen avulla.

ARP-myrkytys Ethernet-kytkimen tai reitittimen tila, jossa tunkeutuja on syöttänyt laitteeseen virheellistä tietoa, saaden sen ohjaamaan liikennettä väärään porttiin. Tätä hyödynnetään välimieshyökkäyksessä.

ATM Alarm Transmitter Module; Telcont Oy:n määrittelemällä signalointiprotokollalla hälytyksiä välittävä asiakaslaite.

ATS Alarm Transmitter Supervisor; palvelinlaite 1-200:lle ATM:lle.

Dynamic C Rabbit Semiconductorin valmistamien moduulien ohjelmankehitysympäristö, jonka käyttämä C-ohjelmointikielen murre poikkeaa merkittävästi ANSI C:stä, joka on American National Standards Instituten määrittelemä, yleisesti käytetty standardi C-kielelle.

Ethernet Eräs laajasti käytetty toteutus OSI-mallin kerroksille 1 ja 2.

Ethernet-kytkin Useita Ethernet-portteja sisältävä laite, joka ohjaa verkkoliikenteen siihen porttiin, josta on reitti haluttuun kohteeseen.

FEP Front End Processor; edustapalvelin. Hätäkeskuslaitoksen palvelin, joka tulkitsee eri yritysten kehittämät hälytyksensiirtoprotokollat hätäkeskuksen laitteiston ymmärtämään muotoon.

Flash-muisti Muistityyppi, joka ei tarvitse virransyöttöä säilyttääkseen sisältönsä, mutta johon kirjoittaminen on hitaampaa ja rajoitetumpaa kuin RAM-muistiin.

Haittakoodi Tunkeutujan syöttämä, yleensä konekielinen ohjelmakoodi, joka tulee ohjelmistossa olevan virheen seurauksena suoritetuksi.

HTML Hypertext Markup Language; kuvauskieli hypertekstiä, eli linkkejä sisältävää tekstiä, kuten internet-sivuja varten.

Integrointi Ohjelman osien saattaminen toimimaan yhdessä, määritellyllä tavalla.

IP-osoitteen väärentäminen (Spoofing attack), datapakettien lähettäjän IP-osoitteen väärentäminen. Menetelmää käytetään tietomurroissa sekä muussa vahingonteossa.

ISDN Integrated Services Digital Network; piirikytkentäinen verkkotekniikka puheen ja datan välittämiseksi puhelinlinjojen välityksellä.

ISJ Ilmoituksensiirtojärjestelmä.

ISP Internet Service Provider; Yritys, jonka välityksellä sen asiakkaat saavat yhteyden Internetiin.

Istunto (sessio) Aikaväli tiettyyn tarkoitukseen luodun datayhteyden alkamisesta sen päättymiseen. Istunnolla voidaan viitata myös itse dataan, joka tänä aikana kulkee.

Ketterä ohjelmistokehitys (agile software development) Vaihtoehtoinen ohjelmistotuotantoprosessi perinteisemmälle vesiputousmallille. Ketterä ohjelmistokehitys on suunniteltu sietämään ohjelmistokehityksen aikana tapahtuvia muutoksia paremmin kuin vesiputousmalli.

MAC Media Access Control; protokolla, joka on osa OSI-mallin kerrosta 2.

MAC-osoite Verkkolaitteen yksilöivä osoite MAC-protokollassa.

Modbus Modiconin kehittämä sarjaliikenneprotokolla, jolla voidaan liikennöidä elektronisten laitteiden välillä. Liikennöinti vaatii isäntälaitteen, johon muut laitteet liittyvät ja joka ohjaa niiden välistä liikennettä.

OSI-malli Open Systems Interconnection Reference Model; seitsenkerroksinen tiedonsiirtoprotokollien kuvaus. Kukin kerros käyttää hyväkseen sitä alempaa kerrosta.

Pakettikytkentäinen verkko Tietoverkko, jossa välitettävä tieto jaetaan pienempiin osiin eli datapaketteihin. Tunnusomaista näille verkoille on, että paketit kulkevat kohteeseensa erilaisia reittejä pitkin, edellisten pakettien reiteistä riippumatta.

Palvelunestohyökkäys Menetelmä, jolla luvattomasti saadaan vähintään yksi tietoverkkoon kytketty laite toimimaan hitaammin tai lopettamaan toimintansa kokonaan.

Piirikytkentäinen verkko Tietoverkko, jossa kutakin istuntoa varten määrätään tietty reitti, jota pitkin tieto kulkee verkossa. Tavallisesti tämä reitti ei vaihdu kesken istunnon.

Porttitoistin (hub) Ethernet-kytkintä muistuttava laite, joka toistaa siihen tietystä portista tulevan verkkoliikenteen kaikkiin muihin portteihin.

PPP Point-to-Point Protocol; protokolla, jolla voidaan muodostaa suora yhteys kahden verkkolaitteen välille esimerkiksi sarjaporttien kautta.

Protokolla (yhteyskäytäntö) Kokoelma sääntöjä, joiden mukaisesti kaksi samaa protokollaa käyttävää laitetta ovat yhteydessä keskenään.

Reititin Ethernet-kytkintä muistuttava laite, jonka läpi kulkevaa liikennettä voidaan hallita tarkemmin kuin kytkimessä.

Signaalin limittäminen (Multipleksaus), menetelmä, jossa useampi signaali välitetään yhtä kanavaa pitkin, esimerkiksi välittämällä yksittäiset signaalit omilla taajuuksillaan.

SRAM Static Random Access Memory; muistityyppi, jonka osoitteiden lukeminen ja kirjoittaminen mielivaltaisessa järjestyksessä on nopeaa, mutta joka vaatii jatkuvaa virransyöttöä säilyttääkseen tietonsa. Muisti ei vaadi muistikennon kondensaattorin säännöllistä virkistämistä kuten samankaltainen DRAM-muisti, mutta vaatii piirilevyltä enemmän tilaa kuin DRAM.

SSL Secure Sockets Layer; protokolla, jolla voidaan estää välimieshyökkäyksen aikana tapahtuva tietoliikenteen analysointi ja muuttaminen. Datapakettien sisältö on voimakkaasti salattu ja istunnon jäsenet pystyvät varmistamaan, että toinen osapuoli on se, jonka hänen oletetaan olevan.

TCP/IP Transmission Control Protocol/Internet Protocol; OSI-mallin kerrokset 3 ja 4 toteuttava protokollien yhdistelmä, joka sisältää perille saapumattomien datapakettien automaattisen uudelleenlähetyksen sekä väärässä järjestyksessä saapuneiden pakettien järjestämisen.

UDP/IP User Datagram Protocol/Internet Protocol; OSI-mallin kerrokset 3 ja 4 toteuttava protokollien yhdistelmä, joka ei varmista datapakettien pääsemistä kohteeseensa tai niiden saapumista lähetysjärjestyksessä.

Vesiputousmalli Ohjelmistotuotantoprosessi, joka etenee ennalta määriteltyjen vaiheiden mukaan, alkaen vaatimusmäärittelyn kirjoittamisesta ja päättyen projektin ylläpitovaiheeseen. Alkuperäisessä vesiputousmallin määritelmässä oli seitsemän vaihetta.

Välimieshyökkäys Murtautumismenetelmä tietojärjestelmiin. Tunkeutuja ohjaa lähiverkossa kulkevan liikenteen hallitsemansa tietokoneen läpi, pystyen näin vaikuttamaan verkkoliikenteeseen kummankaan osapuolen sitä tavallisesti havaitsematta.

Välityspalvelin Lähiverkkoon kytketty laite tai tietokone, jonka läpi kaikki verkosta poistuva ja sinne tuleva liikenne kulkee.

1. JOHDANTO

Tätä työtä aloitettaessa oli ilmoituksensiirrossa alkamassa murroskausi. Ilmoituksensiirto perustui vanhentuvaan tekniikkaan, jossa hälytysten siirtyminen vartiointiliikkeille vaati erikseen siihen tarkoitukseen varattuja kaapeleita. Kuitenkin teknologinen kehityspaine Suomessa oli kohti alati pienenevää määrää maanalaisia kaapeleita. Esimerkiksi internet-operaattorit toivat langattomia yhteyksiä markkinoille. Sittemmin tämän suuntainen kehitys on vain kiihtynyt.

Telcont Oy:ssä päätettiin suunnitella hälytyksensiirtojärjestelmä, joka olisi vapaa vanhenevan tekniikan aiheuttamista rajoitteista, ja kirjoittaja alkoi toteuttaa tähän järjestelmään kuuluvaa asiakaslaitetta. Ajatuksena oli, että asiakkaan olemassa oleva internet-yhteys kelpaisi myös hälytyksensiirtoon, muun käytön ohella. Asiakkaan kustannukset rajoittuisivat siis laitteen hintaan sekä vartiointiliikkeen laskutukseen vartiointipalvelusta.

Seuraavaksi lukijalle annetaan tarvittavaa tietoa ilmoituksensiirtoverkoista, tiettyjen projektissa tehtyjen ratkaisujen selventämiseksi ja kuvaillaan erilaisia tunkeutumismenetelmiä, jotka oli otettava huomioon ja mahdollisuuksien mukaan estettävä. Sulautettuihin järjestelmiin liittyviä asioita, kuten reaaliaikajärjestelmien toimintaa, kuvaillaan tämän jälkeen, ohjelmistoarkkitehtuurin ymmärtämisen helpottamiseksi, mainiten vain työssä käytetyt ominaisuudet. Sen jälkeen kerrotaan projektin lähtökohdat. ATM0602 sekä palvelinlaite ATS muodostavat ilmoituksensiirtojärjestelmän, jonka toiminta oli osittain määritelty ennen projektin alkamista. Lähtökohdat-luvussa kerrotaan näistä asioista.

Suurin osa diplomityöstä koostuu *Työn eteneminen* -luvusta, jossa kerrotaan kirjoittajan osuudesta laitteen suunnittelussa ja toteutuksessa. Tämä työ kuvaa toisaalta ATM0602-moduulin toteutusta, mutta toisaalta se on myös esimerkki ketterästä projektista. Siksi luvussa on kiinnitetty erityistä huomiota työn ajalliseen etenemiseen.

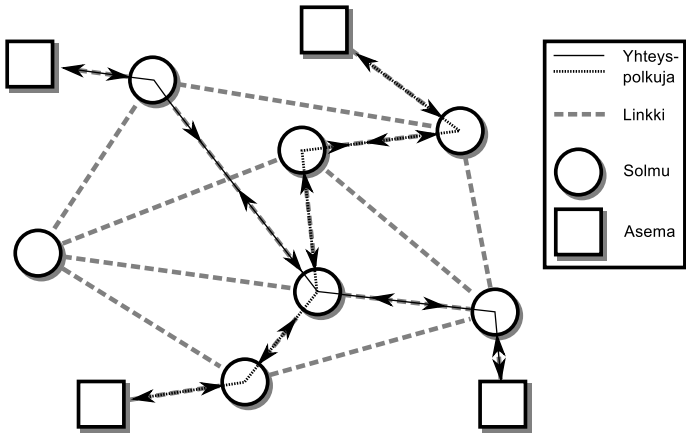
Muiden Telcont Oy:n työntekijöiden toteuttamista ohjelman osista puhutaan lyhyesti sikäli, kuin se on olennaista kirjoittajan tekemän ohjelmistoon integroinnin kannalta. Lopuksi arvioidaan projektin onnistumista ja mainitaan muutamia, projektin aikana opittuja asioita.

2. VERKKOTEKNIIKAT JA VANHAT HÄLYTYSVERKOT

Aikaisemmat ilmoituksensiirtomenetelmät perustuivat piirikytkentäisten verkkojen hyödyntämiseen. ATM0602 taas edustaa siirtymistä pakettikytkentäisiin verkkoihin. Tämä siirtymä on jo tapahtunut tai on tapahtumassa useimmilla muilla tiedonsiirron osa-alueilla. VoIP, eli Voice over Internet Protocol, alentaa puhelinyhteyksien kustannuksia kuljettamalla puheen datapaketteina alusta loppuun saakka. ADSL on jo aikaa sitten syrjäyttänyt ISDN-yhteydet tarjoamalla nopeampia Internet-yhteyksiä alemmalla hinnalla. Pakettikytkentäisyys tuo huomattavia säästöjä kustannuksissa, mutta helpottaa myös luvatonta tietoverkkoihin tunkeutumista. Tätä ongelmakenttää käsitellään niin ikään tässä luvussa.

2.1 Piirikytkentäiset verkot

Kun piirikytkentäisen verkon asiakas pyytää yhteyttä toiseen asiakkaaseen, luodaan istunnon ajaksi muuttumaton reitti verkon läpi. Istunto päättyy, kun toinen asiakkaista katkaisee yhteyden. Reitti kulkee verkon solmusta toiseen solmuun näiden välillä olevien linkkien kautta. Kuvassa 2.1 on esimerkki piirikytkentäisestä verkosta, jossa on muodostettu kaksi polkua.



Kuva 2.1: *Esimerkki piirikytkentäisestä verkosta, jossa on muodostettu kaksi yhteyspolkua*

Ensimmäisissä puhelinverkoissa asiakas soitti aina ensin puhelinvaihteeseen. Se oli verkon solmu, ja linkkejä olivat sähköiset yhteydet kuhunkin verkon asiakkaan puhelimeen. Myös puhelinvaihteiden välillä oli linkkejä, ja kaukopuhelut kulkivat useamman solmun kautta. Piirikytkentäisessä verkossa kulutetaan tarpeettomasti resursseja. Ensimmäisissä verkoissa kukin yhteyspolku varasi kaikki käyttämänsä linkit istunnon ajaksi. Myöhemmin saatiin vietyä useampia kanavia linkkien läpi esimerkiksi hyödyntämällä eri taajuusväliä kullekin kanavalle. Tällöin yhteyspolku varasi koko linkin sijasta yhden sen kanavista.

Monikanavaisuus ei ole tehokas ratkaisu kahdesta syystä. Ensiksikin kanava on varattuna, vaikka sen läpi ei kulkisi lainkaan tietoa. Puhelinyhteyksissä tämä on harvoin ongelmallista, mutta vaikkapa Internetin selaamisessa se on. Useimmiten asiakkaan lukiessa tiettyä sivua hänen ei tarvitse siirtää dataa, mutta tästä huolimatta modeemi- ja ISDN-yhteydet kasvattavat puhelinlaskua siltä ajalta. Toinen ongelma liittyy verkon rakenteeseen. Jotkut linkit ovat useammin käytössä kuin toiset, ja käyttöaste vieläpä vaihtelee eri aikoina. Tämän vuoksi kukin linkki on mitoitettava suurimman käyttöasteen mukaan. Yhteyspolun muodostamiseen kestää myös aina jonkin verran aikaa.

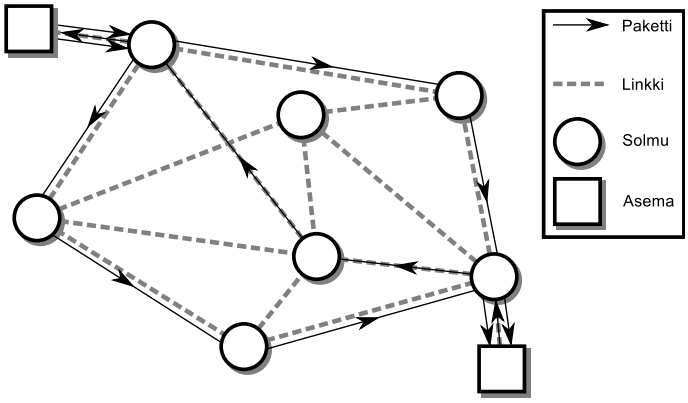
2.2 Pakettikytkentäiset verkot

Verkon kuormitus jakautuu tasaisemmin, kun käytetään pakettikytkentäistä tekniikkaa. Tällöin datan saapuessa verkkoon se jaetaan pienempiin osiin, joita kutsutaan datapaketeiksi. Niihin lisätään myös otsikot, jotka ilmaisevat vähintään sen, minne paketti on menossa ja kuinka paljon dataa se sisältää.

Useimmiten otsikkoon lisätään myös lähetysosoite sekä tarkistussumma, jonka avulla voidaan havaita tiedonsiirrossa mahdollisesti tapahtuneet virheet. Pakettikytkentäisen verkon etu on se, että paketit voidaan välittää kohteeseensa eri polkuja pitkin. Jos jokin pakettien käyttämä linkki ylikuormittuu istunnon aikana, voidaan seuraavat paketit kuljettaa pidempää mutta vähemmän kuormitettua reittiä.

Seuraavalla sivulla olevassa kuvassa 2.2 on pakettikytkentäinen verkko, jonka rakenne on sama kuin aiemmin esitetyn piirikytkentäisen verkon. On myös mahdollista muodostaa virtuaalipiiri pakettikytkentäisessä verkossa, jolloin paketit kulkevat piirikytkentäisen verkon tapaan samaa polkua.

Sama palvelunlaatu voidaan siis pakettikytkentäisessä verkossa saavuttaa pienemmillä sijoituksilla. Asiakkaille tämä näkyy alempana hintana sekä mahdollisuutena laskutukseen kiinteällä kuukausimaksulla, sen sijaan että laskutettaisiin lähetetyn ja vastaanotetun datan määrän mukaan. Pii-



Kuva 2.2: *Esimerkki pakettikytkentäisestä verkosta, jossa kahden verkon aseman välillä kulkee kolme datapakettia*

rikytkentäisessä verkossa yhteyden muodostamiseen kuluu myös jonkin verran aikaa. Pakettikytkentäisessä verkossa laite voi olla jatkuvasti yhteydessä, jolloin tästä johtuvaa viivettä ei ole. Koska kanavaa ei tarvi pitää tällöinkään varattuna, jatkuva yhteys ei aiheuta ylimääräisiä kuluja, jotka lopulta laskutettaisiin asiakkaalta.

2.2.1 Pakettikytkentäisyyden haittapuolet

Koska verkon solmujen kannalta kaikki datapaketit ovat itsenäisiä, eivätkä liity millään tavalla aiemmin lähetettyihin paketteihin, ne saattavat saapua kohteeseensa eri järjestyksessä kuin ne lähetettiin. Yksi tapa välttyä tältä on odottaa että kohde vahvistaa ottaneensa paketin vastaan lähettämällä toisen paketin, ja vasta tämän jälkeen lähetetään seuraava. Tämä on kuitenkin epäkäytännöllistä jos suuri määrä tietoa on siirrettävä nopeasti. Useimmiten tosin verkossa siirretään paketit virtuaalipiirien avulla, jolloin tätä ongelmaa ei esiinny. Jollei asiakas ole kuitenkaan erikseen tilannut virtuaalipiiriä, voi palveluntarjoaja muuttaa reititysmenetelmää milloin tahansa. [19, s. 308]

Toinen ongelma on, että jotkut paketeista eivät välttämättä saavu edes perille. Polut kulkevat usein niin monen solmun läpi, ettei voida taata jokaisen onnistuvan tehtävässään joka kerta. Kadonneiden pakettien määrä vaihtelee verkon kuormitusasteen mukaan. Nämä kaksi ongelmaa on huomioitava kaikissa sovellutuksessa, jotka hyödyntävät pakettikytkentäistä verkkoa.

2.2.2 UDP

IP (Internet Protocol) on OSI-mallin kerroksen kolme protokolla. Sen tarkoitus on saada kuljetettua datapaketti haluttuun osoitteeseen, jota kutsutaan

IP-osoitteeksi. Protokolla ei kuitenkaan takaa paketin pääsyä perille tai pakettien oikeaa järjestystä. Paketti saattaa myös saapua useampana kappaleena ja sen datassa voi olla virheitä. IP varmistaa ainoastaan, että paketin otsikossa tapahtuneet virheet havaitaan.

UDP (User Datagram Protocol) tarjoaa minimimäärän palveluja OSI-mallin neljännen kerroksen vaatimusten toteutumiseen. Se lisää IP:hen ainoastaan porttiosoituksen. Näin kukin IP-osoite voidaan jakaa yhteensä 65536:een porttiin. Tällöin tiedetään otsikon perusteella, minkä sovelluksen tulee käsitellä paketti. UDP ei siis vaikuta IP:n luotettavuuteen. Sitä käyttävän sovelluksen, joka edustaa OSI-mallin kerrosta seitsemän, tulee tarvittaessa toteuttaa luotettavuuteen liittyvät palvelut.

2.2.3 TCP

TCP (Transmission Control Protocol) on toinen OSI-mallin neljännen kerroksen protokolla UDP:n lisäksi. TCP sisältää porttiosoituksen ja toteuttaa lisäksi viisi luotettavuusvaatimusta. Pakettien pääsy perille varmistetaan ja paketti lähetetään tarvittaessa uudelleen. Väärässä järjestyksessä saapuneet paketit järjestetään ennen niiden välittämistä sovellukselle, ja monistuneista paketeista välitetään vain yksi. Mikäli dataan on tullut virheitä, pyydetään lähettäjää välittämään paketti uudelleen. Jos taas paketteja saapuu nopeammin kuin niitä ehditään käsitellä, pyydetään lähettäjää hidastamaan viestinvälitystä.

TCP/IP on hyvä yhteyskäytäntö, kun on välttämätöntä saada jokainen paketti perille. On kuitenkin reaaliaikasovelluksia, joissa voidaan hyväksyä se, että osa paketeista katoaa, ja tällöin vältetään uudelleenlähetyksistä aiheutuvat viiveet. TCP/IP:llä välitetyn paketin otsikko on myös pidempi kuin UDP:ssä, joten sovelluksen kannalta hyödyllistä tietoa saadaan mahtumaan vähemmän samaan kaistanleveyteen.

Vakavampi ongelma on kuitenkin se, että TCP/IP ei sisällä menettelyjä tietoturvan varmistamiseen. Näitä puutteita hyväkseen käyttävät menetelmät tietojärjestelmiin tunkeutumiseen ja muuhun vahingontekoon ovat yleisesti tunnettuja ja niistä löytyy tietoa esimerkiksi julkisista kirjastoista. Tavanomaisia ratkaisuja tähän ovat erilaiset salausmenetelmät kuten SSL (Secure Sockets Layer).

Tämän projektin kannalta merkittävä menetelmä on SYN-tulva, joka tapahtuu seuraavanlaisesti: Tunkeutuja lähettää suuren määrän SYN-viestejä hyökkäyksen kohteelle, useista IP-osoitteista. Ne voivat olla väärennettyjä osoitteita tai kuulua tietokoneille, jotka tunkeutuja on saanut vaikkapa haittaohjelmien avulla haltuuna. SYN-viesti merkitsee uuden yhteyden avaa-

mista, ja tälle yhteydelle varataan hyökkäyksen kohteessa resursseja. Suuri määrä tällaisia viestejä kuluttaa kohteen TCP-protokollapinon resurssit loppuun, eikä kohde ehdi palvella muiden laitteiden lähettämiä SYN-viestejä. Näin kohteeseen ei saada enää yhteyttä, ja palvelunestohyökkäys on onnistunut. [15, s. 3–5]

2.3 Tietoturva pakettikytkentäisissä verkoissa

Tietoturvaongelmat saavat yleensä alkunsa käyttöjärjestelmästä, sen palveluista sekä käynnissä olevista sovelluksista. Toisinaan tunkeutujat soluttautuvat myös fyysisesti yrityksen tiloihin esiintyen vaikkapa konsultteina tai jopa hakevat työpaikkaa yrityksestä vain tehdäkseen tietomurron. Kuitenkin tavallisin tilanne on, että tunkeutuja käyttää hyväkseen jotain käyttöjärjestelmän tai sovelluksen haavoittuvuutta ja saa siten asennettua oman tietokoneohjelmansa Internetin välityksellä johonkin yrityksen tietokoneeseen.

On olemassa helposti saatavilla olevia tietokoneohjelmia, jotka etsivät yritysverkosta haavoittuvia tietokoneita. Tunkeutujan ei tarvitse kuin tietää jonkin yritysverkon IP-osoite. Kun sopiva tietokoneohjelma on saatu asennettua, tietokone on tunkeutujan hallinnassa. Mikäli tietokoneesta onnistutaan vaihtaa käyttöjärjestelmän ydin muokatulla versiolla, on myös mahdollista peittää jälkensä täydellisesti. Ydintä voidaan muokata niin, että se käsittelee järjestelmän antamia vasteita ikään kuin mitään muutoksia ei olisi tehty. Esimerkiksi tiedostolistauksesta voidaan poistaa tunkeutujan lisäämät tiedostot ennen kuin käyttäjä näkee listauksen. Ainoastaan tietokoneen käynnistäminen toiselta massamuistilta, jossa on muokkaamaton käyttöjärjestelmä, voisi paljastaa tietomurron tapahtuneen.

2.3.1 Porttiskannaus

Yleisin tietoturvauhka alkaa porttiskannauksesta. Tällöin tietyn IP-osoitteen kuhunkin porttiin koetetaan ottaa yhteyttä TCP/IP:llä sekä UDP:lla. Tietokone antaa yhteydenottoon vastauksen ICMP-protokollaa käyttäen, mikäli tätä ei ole erikseen estetty palomuurissa tai käyttöjärjestelmässä. Jos mikään sovellus tai palvelu ei käytä kyseistä porttia, lähetetään tästä kertova ICMP-vastaus yhteydenottajalle. Tämä voi jo antaa tunkeutujalle hyödyllistä tietoa, koska eri käyttöjärjestelmissä ja jopa niiden eri versioissa on pieniä eroja sen suhteen, miten vastaus on muotoiltu. [13, luku 2, s. 12] Kun tunkeutuja saa selville tietokoneen käyttöjärjestelmän, hän voi käyttää hyväkseen sen tunnettuja tietoturva-aukkoja.

Vielä enemmän tietoa kohteen tietoturva-aukoista saadaan silloin, kun löydetään portti, johon on liitetty sovellus tai palvelu, eli portti on avoinna. Tällöin järjestelmä antaa uudenlaisen vastauksen, ja yhdistelemällä saatuja tietoja vastausten tarkasta muodosta, saadaan suljettua pois käyttöjärjestelmäversioita, joiden joukosta kohteen todellinen versio löytyy. Tätä menetelmää jatkamalla saadaan usein selville tarkka käyttöjärjestelmäversio, tai ainakin pieni määrä eri mahdollisuuksia, joiden tunnettuja tietoturva-aukkoja voidaan sitten järjestyksessä yrittää käyttää hyväksi. On olemassa ohjelmia, jotka tekevät tämän automaattisesti. Kun tällaiselle ohjelmalle syöttää IP-osoitteita, ohjelma kertoo minkä käyttöjärjestelmän, tai mahdollisia käyttöjärjestelmiä, se kustakin osoitteesta selvitti.

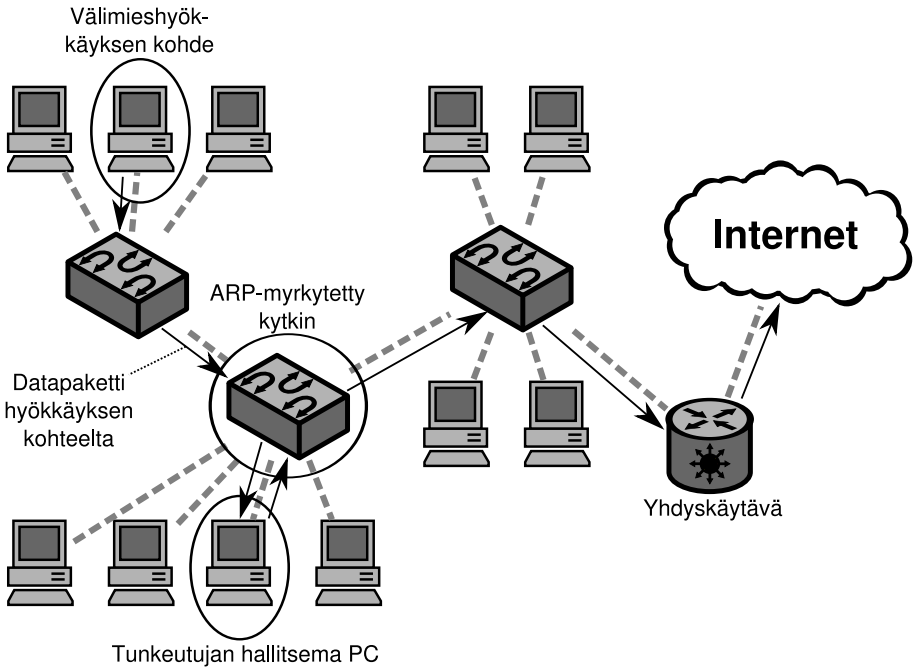
TCP/IP on UDP:tä ongelmallisempi protokolla tietoturvan kannalta. Avoinna oleva UDP-portti voidaan ohjelmoida vastaamaan ainoastaan tarkalleen oikealla tavalla muotoiltuun viestiin. Tällöin tunkeutujan on vastauksen saadakseen tiedettävä, mikä ohjelma tai palvelu porttiin on liitetty ja miten sen kanssa liikennöidään. TCP/IP taas sisältää kättelymenettelyn, joka edeltää liikennöintiä. Tällöin riittää aloittaa kättely kyseiseen porttiin, jotta saataisiin tietää sen olevan avoinna. Kättelymenettelyn yksityiskohdat voivat myös antaa tietoa kohteen käyttöjärjestelmästä.

Jos yritysverkossa on yksikin palvelin, joka antaa vastauksia yritysverkon ulkopuolelle, tunkeutuja saa sen perusteella selville paljon muitakin yritysverkon osoitteita. Osoitteet kun ovat lähes aina vierekkäisiä numeroita, eli IP-osoitteen neljästä numerosta vain viimeinen muuttuu. Siksi tietoturvan kannalta kriittistä sovellusta tai laitetta suunniteltaessa on aina lähdettävä siitä oletuksesta, että tunkeutuja saa kohdistettua siihen porttiskannauksen, vaikka hänellä olisi pelkästään julkisesti saatavilla olevaa tietoa sen omistavasta yrityksestä. Erityisen helppoa osoiteavaruuden selvittäminen on, jos yrityksen kotisivut ovat yritysverkossa olevalla palvelimella.

Myös yksityishenkilö voi joutua tällaisen hyökkäyksen kohteeksi, koska tietomurtoja harrastavat henkilöt tekevät usein porttiskannauksia sattumanvaraisiin IP-osoitteisiin. Tunkeutujalle ennestään tuntematon laite tai sovellus saatetaan IP-osoitteesta paljastuessaan kokea haasteena.

2.3.2 Välimieshyökkäys

Kun tietoverkkoon tunkeutuminen on tapahtunut, on yritysverkossa mahdollista suorittaa välimieshyökkäys. Ethernet-kytkimet, jotka ovat yritysverkon solmuja, pitävät yllä taulukkoa siitä, mikä MAC-osoite vastaa mitäkin IP-osoitetta. Kun tunkeutuja lähettää kytkimelle väärennetyn ARP-vastauksen kaapatusta tietokoneesta, hän saa kytkimen yhdistämään haluamansa IP-



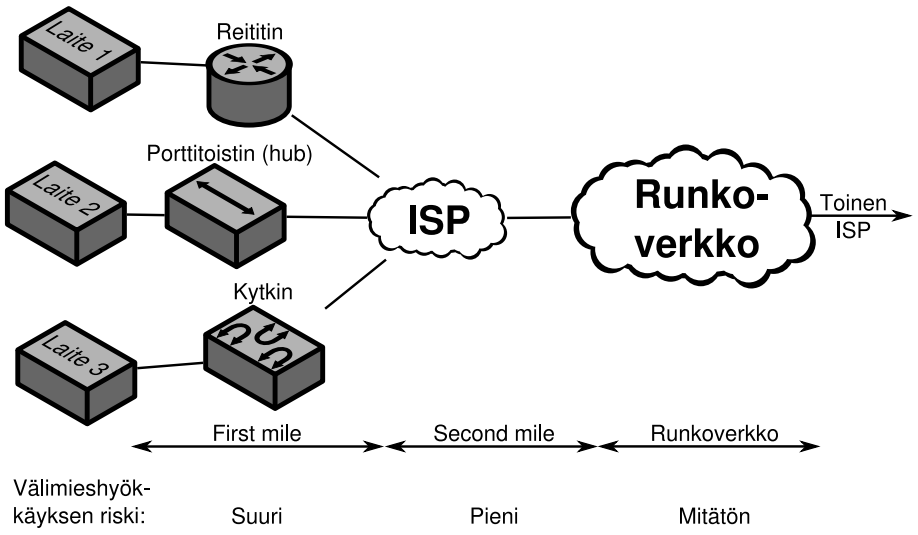
Kuva 2.3: *Välimieshyökkäys yritysverkossa*

osoitteen kaapatun tietokoneen MAC-osoitteeseen, jolloin liikenne ohjautuu sinne. Tätä kutsutaan kytkimen ARP-myrkyttämiseksi.

ARP-myrkyttämistä käyttävä tunkeutuja voi tarkkailla ja muutella paketteja, ja lähettää ne sitten eteenpäin oikeaan kohteeseensa. Jollei käytössä ole turvalliseksi suunniteltua protokollaa, kuten SSL:ää, kumpikaan välimieshyökkäyksen kohteista ei havaitse hyökkäystä. Esimerkki välimieshyökkäyksestä yritysverkossa on kuvassa 2.3.

Omassa yritysverkossa tapahtuva välimieshyökkäys on mahdollista estää käyttämällä kytkimien sijaan reitittimiä. Niissä on mahdollista hallita tarkemmin niitä sääntöjä, joiden mukaan paketteja käsitellään. Reitittimelle voidaan rajapintakohtaisesti määritellä pääsylista, jolla voidaan sallia liikennöinti vain tietyistä IP-osoitteista. [20, s. 165] Yritysverkon rakenne voidaan näin ohjelmoida reitittimiin käsin, jolloin ne eivät huomioi väärennettyjä ARP-vastauksia, jotka saapuvat kielletystä IP-osoitteesta. Reitittimet ovat kuitenkin kytkimiä kalliimpia, ja yritysverkon rakenteen muuttaminen jälkeenpäin vaatii tätä menettelyä käytettäessä aina muutosta yhteen tai useampaan reitittimeen. On toisaalta mahdollista myös määritellä vain osa IP-osoitteista käsin, jos tiedetään että tietyissä reiteissä tapahtuva välimieshyökkäys on vakavampi uhka kuin toisissa.

IP-osoitteiden käsin määritteleminen ei takaa sitä, etteikö hyökkäystä voitaisi suorittaa istunnon toisen osapuolen verkossa. Myös internet-



Kuva 2.4: *First mile, second mile sekä runkoverkko*

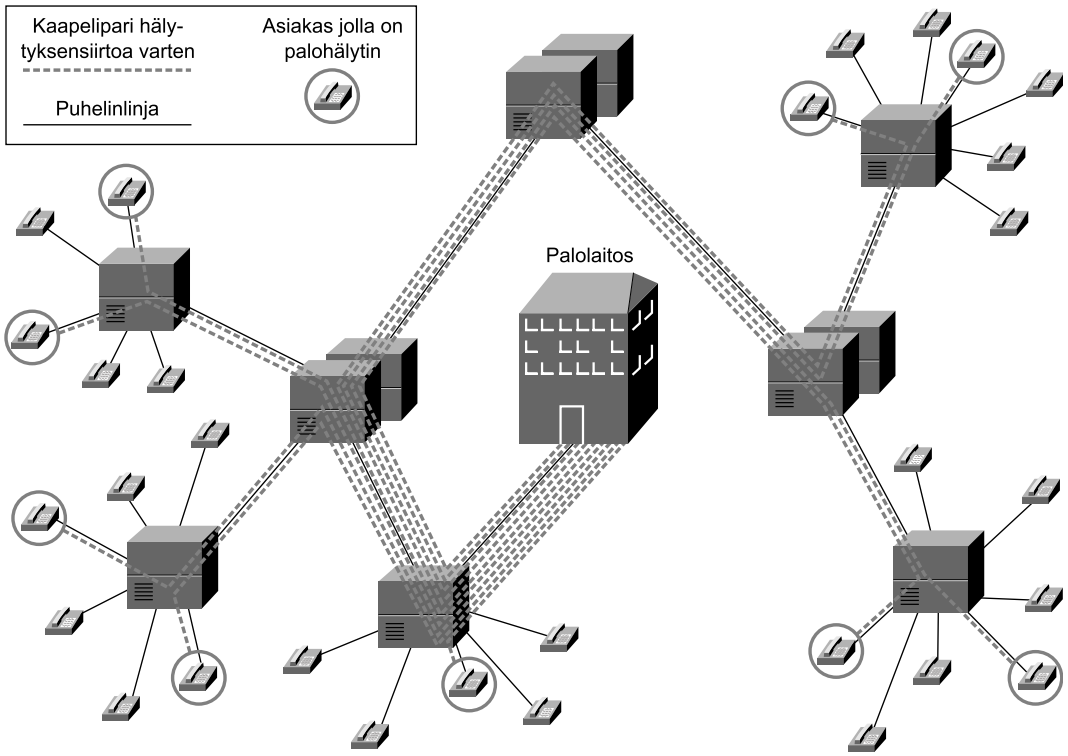
palveluntarjoajan eli *ISP*:n verkkoon tunkeutuminen on mahdollista, vaikkakin hankalampaa. Runkoverkkoon tunkeutuminen on niin vaikeaa, että tämä uhka voidaan jättää huomiotta, koska runkoverkkojen solmut ovat tarkasti vartioituja ja useimmiten suuryritysten tai valtion omaisuutta.

Paketin reittiä yritysverkon tietokoneesta ISP:lle kutsutaan first mile -yhteydeksi. Vastaavasti reitti ISP:ltä runkoverkkoon on nimeltään second mile. Näitä käsitteitä ja niihin liittyvää välimieshyökkäyksen riskiä esitetään kuvassa 2.4.

2.4 Olemassa oleva hälytyksensiirtotekniikka

Ensimmäiset Suomessa käytetyt ilmoituksensiirtojärjestelmät oli tarkoitettu ainoastaan paloilmoituksiin. Ne noudattivat puhelinverkkojen topologiaa. Ne perustuivat suoriin kaapelipareihin valvottavasta kohteesta palolaitokseen puhelinverkon pääte- ja solmukeskuksien kautta seuraavalla sivulla olevan kuvan 2.5 mukaisesti. Kohteiden määrän kasvaessa tämä kävi epäkäytännölliseksi ja kalliiksi.

Ongelma ratkaistiin korvaamalla kaapeliyhteydet puhelinverkossa kulkevilla signaaleilla, jotka limitettiin puheeseen käytetyn taajuusalueen yläpuolella. Tällöin kaapelin sijasta tarvittiin ainoastaan lähetin-vastaanotinpari sekä tavallinen puhelinlinja kutakin kohdetta varten. Näin asiakkaan oli mahdollista käyttää puhelintaan samaan aikaan kuin ilmoituksensiirtolaitteet olivat yhteydessä niitä valvovaan tahoon. Järjestelyssä oli kuitenkin se ongelma, että erityisesti solmukeskuksien välillä kulki niin suuri määrä yhteyksiä, että niiden välisen puhelinlinjan katkeaminen olisi tuhoisaa. Sik-

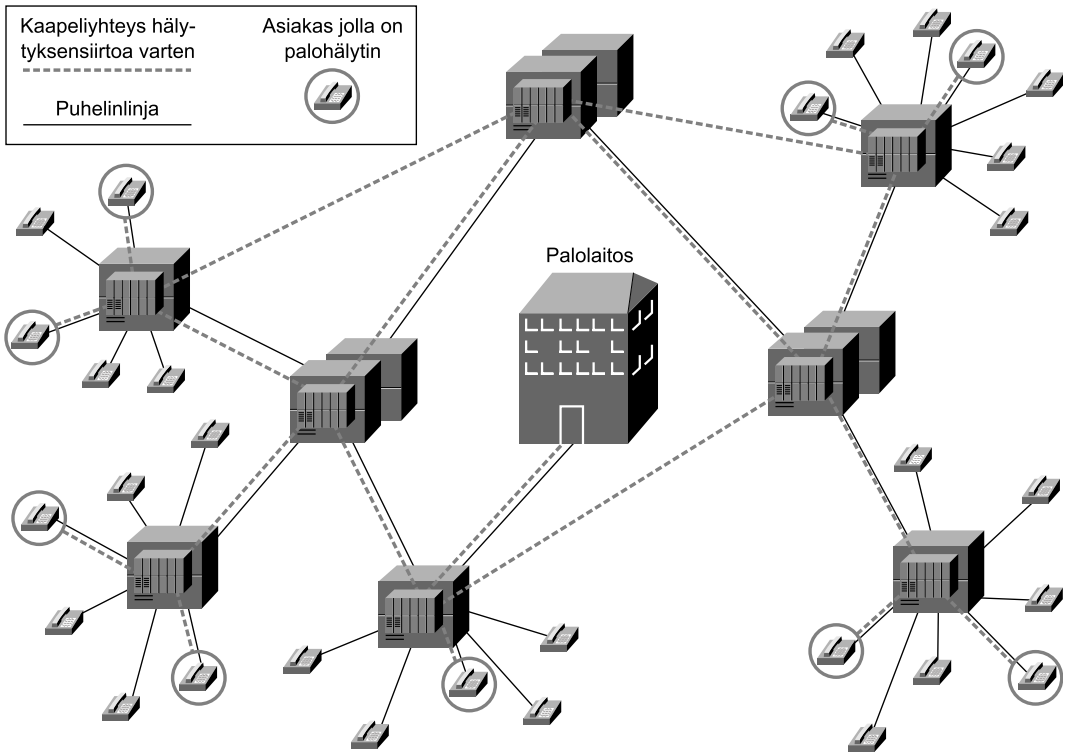


Kuva 2.5: *Palohälytykset ensimmäisissä hälytysverkoissa [8, kalvo 1]*

si puhelinverkkoon lisättiin varayhteyksiä solmukeskusten välille, niin että hälytyksellä olisi aina vaihtoehtoinen reitti, jos yksi yhteys katkeaisi, kuten seuraavalla sivulla olevasta kuvasta 2.6 ilmenee. Nykyinen hälytyksensiirtoverkko on siis tavanomainen, aliluvussa 2.1 kuvattu piirikytkentäinen verkko.

Muun muassa vesilaitokset, jätevesilaitokset, vartiointiliikkeet, sähkölaitokset ja kiinteistöyritykset kiinnostuivat myös kaukovalvonnasta. Valvottavien kohteiden määrä uhkasi siis jälleen moninkertaistua, ja mikä pahempaa, samaa verkkoa käyttäisivät toisistaan riippumattomat ja tietämättömät tahot. Oli välttämätöntä suunnitella hierarkkinen verkko, jotta vältyttäisiin ristiriitaisuuksilta, kuten saman tunnisteen esiintymiseltä kahden eri organisaation kohteessa.

Hierarkkisen verkkoratkaisun nimeksi tuli ComNet ja sen pääkehittäjä oli Computec Oy. Se oli monisuuntainen datayhteys, joka kehitettiin jo ennen internetin keksimistä. Verkon hierarkia oli kolmitasoinen. Kukin verkon osoite oli 32-bittinen luku, joka koostui kahdeksanbittisestä yläverkosta, kahdeksanbittisestä alaverkosta ja 16-bittisestä pisteosoitteesta. Ylä- ja alaverkkonumerot olivat välillä 1–99. Yläverkkoja jaettiin organisaatiokohtaisesti. Alaverkot sisälsivät 65535 pisteosoitetta. Kukin pisteosoite merkitsi yhtä häly-

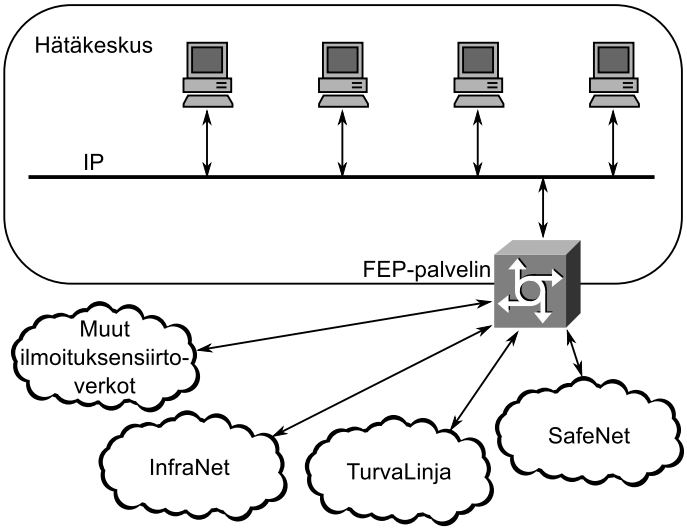


Kuva 2.6: *Nykyinen hälytyksensiirtotekniikka, jossa signaalit on limitetty ja osa niistä varmistettu varayhteyksillä [8, kalvo 7]*

tyssilmukkaa, sensoria, etäohjattavaa relettä tai muuta vastaavaa kohdetta. Näin voitiin taata, että eri organisaatioiden kohteilleen jakamat pisteosoitteet pysyivät erillisinä.

ComNetin osoitteiden 16-bittisyys asettaa rajoituksia hälytyksensiirtotekniikan uudistamiselle. Tekniikka on kehitetty 70-luvulla ja laitteiden on vielä nykyäänkin toimittava sen asettamien rajoitusten mukaisesti. Ylä- ja alaverkkoja voisi esimerkiksi muuten olla 255 kappaletta, mutta yksikin hälytyksen reitillä oleva laite, joka olettaa, että luvun on oltava alle sata, pysäyttäisi hälytyksen kulun. Lisäksi olisi luontevaa että pisteosoitteillakin olisi kaksitasoinen hierarkia, jossa ylempi kerros merkitsisi tiettyä rakennusta ja alempi sen tiettyä hälytyssilmukkaa. Nykytilanteessa hätäkeskuksen on taulukoitava katuosoite kutakin pistettä eli hälytyssilmukkaa kohti, mikä merkitsee moninkertaista työmäärää osoitteen muuttuessa.

Usea yritys kehitti omia ilmoituksensiirtojärjestelmiään yhtäaikaa. Esimerkiksi Telecom Finland Oy:n järjestelmä oli nimeltään SafeNet ja Computec Oy:n oli TurvaLinja, joka perustui ISDN-tekniikkaan. Ei olisi ollut käytännöllistä muuttaa hätäkeskuslaitoksen tietojärjestelmiä tukemaan jokaista olemassa olevaa, toisistaan poikkeavaa yhteyskäytäntöä, joten kehitettiin



Kuva 2.7: *Hätäkeskuslaitoksen tietojärjestelmä*

FEP eli edustapalvelin. Se tulkitsee eri yhteyskäytännöt hätäkeskuksen tietojärjestelmän ymmärtämään muotoon. Toisaalta se asettaa myös rajoituksia siirrettävälle tiedolle, koska liityntärajapinta on edelleen lähes alkuperäisessä muodossaan. Yhtä hälytystä merkitsevät sanomat ovat pituudeltaan noin 50 bittiä. Tämä johtuu siitä, että tuohon aikaan hitaimmat FEP-palvelinta käyttävät järjestelmät liikennöivät kahdensadan baudin nopeudella. Hätäkeskuksen tietojärjestelmä on esitettynä kuvassa 2.7.

Tämän aliluvun kuvat on mukailtu lähteestä [8, kalvot 2, 5, 7 ja 16] ja muutkin esitetyt tiedot ovat peräisin samasta esityksestä. Esityksen laatija ja Telcont Oy:n perustaja Jussi Hakulinen on ollut mukana toteuttamassa tässä kuvattuja ilmoituksensiirtojärjestelmiä.

3. TYÖHÖN LIITTYVÄÄ OHJELMISTOTEKNISTÄ TAUSTATIETOA

ATM0602-moduulin suoritin oli jo valittu, kun ohjelmiston suunnittelu aloitettiin. Kehitysympäristöön oli saatavilla sille suunnattu versio eräästä suositusta reaaliaikaympäristöstä. Se oli irrottavaa tyyppiä, joten tässä luvussa kuvataan sellaisen reaaliaikajärjestelmän toimintaa ja erityispiirteitä.

Koska kyse on hälytyksensiirtojärjestelmästä, on myös otettava huomioon, että virheellisellä toiminnalla voi olla vakavia seurauksia. Siksi oli keskityttävä erityisen paljon myös ohjelmiston haavoittuvuuksien ehkäisemiseen. Tämä luku sisältää tietoa niistä yleisistä haavoittuvuuksista, jotka ovat olennaisia ohjelmiston kannalta. Esimerkiksi käyttöjärjestelmiin ja niiden eri palveluihin liittyvät haavoittuvuudet sivuutetaan, koska käytetty reaaliaikaympäristö ei tarjoa sellaisia palveluja, joita menetelmät käyttävät hyväkseen. Tämän reaaliaikaympäristön toiminta määräytyy käännösvaiheessa, eikä ympäristöön ole ajonaikaista rajapintaa. Mukana ei myöskään ole tiedostojärjestelmää.

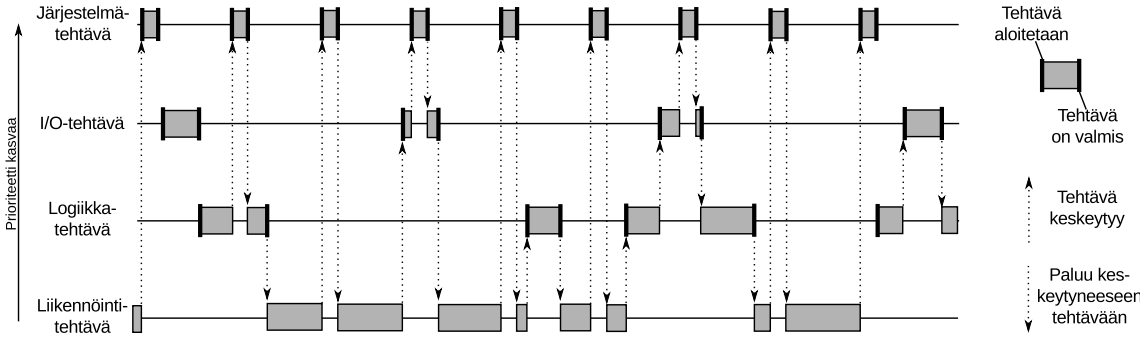
3.1 Irrottavat reaaliaikajärjestelmät

Reaaliaikaympäristö on käyttöjärjestelmä tai ohjelmakirjasto, joka mahdollistaa useiden tehtävien suorittamisen näennäisen yhtäaikaisesti. Sitä käyttävä laite tai järjestelmä on nimeltään reaaliaikajärjestelmä. Jos laitteessa on vain yksi kappale yksiytimisiä suorittimia, ei useampi tehtävä voi olla suorituksessa yksittäisellä ajanhetkellä. Reaaliaikaympäristön tehtävä on ohjata sitä, milloin kukin tehtävä pääsee suoritukseen. Se myös tarjoaa palveluja viestien välittämiseksi tehtävien kesken. Joskus tehtävien kesken tapahtuu poikkeustilanteita, kuten silloin, jos yksi tehtävä ei saa lukea toisen tehtävän tuottamaa tietoa silloin, kun tiedon muodostus on kesken. Reaaliaikajärjestelmä tarjoaa palveluja tällaisten ongelmatilanteiden ratkaisuun.

Irrottava reaaliaikaympäristö päästää suoritukseen aina sen tehtävän, jolla on korkein prioriteetti niistä tehtävistä, jotka eivät ole odottamassa. Samaa prioriteettinumeroa ei valitussa ympäristössä voida antaa useammalle tehtävälle. Irrottava reaaliaikajärjestelmä tarkoittaa sitä, että jos parhail-

laan suoritettavaa tehtävää korkeammalla prioriteetilla oleva tehtävä lopettaa odottamisen, ensiksimainittu keskeytetään ja korkeampiprioriteettinen tehtävä siirretään suoritukseen. Kun tämä tehtävä sitten siirtyy odottamaan, palautetaan keskeytetty tehtävä suoritukseen. Kaikkien tehtävien on myös mahdollista olla odottamassa tietyllä hetkellä. Tällöin ensimmäinen suoritettavaksi pyrkivä tehtävä pääsee heti suoritukseen.

Tapahtumahetkien takarajat liittyvät läheisesti reaaliaikajärjestelmiin. Takarajaa kutsutaan kovaksi, jos sen ylittäminen johtaa vakavaan virheeseen. Esimerkkinä annettakoon nykyaikaisen auton ohjausjärjestelmä. Jos polttoaineensyöttö tapahtuu liian myöhään, se heikentää moottorin tehoa enemmän, kuin jos sitä ei olisi tapahtunut ensinkään, koska sylinteri on jo liikkumassa toiseen suuntaan. Pehmyt takaraja taas tarkoittaa, että ylitys aiheuttaa ainoastaan vähäistä haittaa, ja harvakseltaan tapahtuvat ylitykset voidaan sallia. [11, s. 2]



Kuva 3.1: *Esimerkki irrottavan reaaliaikajärjestelmän toiminnasta*

Kuvassa 3.1 esitetään irrottavan reaaliaikajärjestelmän toimintaperiaate. Kuva on mukailtu ATM0602-moduulin ohjelmiston toiminnasta, mutta suoritusaikoja on liioiteltu. Tässä tapauksessa alimman prioriteetin tehtävä ei koskaan siirry odottamaan kuten todellisessa ohjelmistossa. Mukana on myös järjestelmätehtävä, jonka toiminnan määrittelee reaaliaikajärjestelmä itse. Se hallitsee muita tehtäviä, ja on automaattisesti ylimmällä prioriteetilla.

3.1.1 Reaaliaikajärjestelmiin liittyviä ongelmia

Irrottavan reaaliaikajärjestelmän toimintaperiaatteella on hyviä ja huonoja puolia. Etuna on ennustettavuus. On mahdollista piirtää aikajanalle tehtävien suoritusjärjestys ja esimerkiksi todistaa, että pahimmassakin tapauksessa jokin aikakriittinen asia saadaan valmiiksi ennen tiettyä takarajaa.

Todistuksen oikeellisuus riippuu tietenkin siitä, että piirretyksi todellakin tulee pahin tapaus. Tässä työssä kuitenkin rakennettiin pehmeän takarajan reaaliaikajärjestelmä, joten formaaleja menetelmiä ei käsitellä tarkemmin.

Irrottavalla toiminnalla on myös vakava haittapuoli. Jos jokin alinta prioriteettiä korkeammalla olevista tehtävistä jää loputtomaan silmukkaan, eikä siten koskaan siirry odottamaan, ei myöskään sitä alemman prioriteetin tehtävät pääse koskaan suoritukseen. Ohjelmiston toiminta voi myös viivästyä vakavasti, jos tapahtuu suunnitteluvirheestä johtuva käänteisprioriteettiongelma, eli englanniksi priority inversion. Se tarkoittaa, että alhaisen prioriteetin tehtävä lukitsee tietyn resurssin itselleen ja sen jälkeen korkeamman prioriteetin tehtävä pääsee suoritukseen, tarviten samaa resurssia. Tavallisesti korkeamman prioriteetin tehtävä vain siirtyisi odottamaan havaitessaan resurssin lukituksi. Alemman prioriteetin tehtävä lopulta vapauttaisi resurssin. Jos kuitenkin kolmas tehtävä, joka on prioriteetiltaan näiden välissä, pääsee suoritukseen, joutuu alimman prioriteetin tehtävä odottamaan sen suoritusta loppuun asti. Tällöin resurssi ei vapaudu, eikä myöskään ylimmän prioriteetin tehtävä pääse suoritukseen, vaikka sen oli tarkoitus voida keskeyttää prioriteetiltaan keskimmäisen tehtävän suoritus.

Jos toisaalta voidaan hyväksyä se, että laitteen ohjelma saattaa käynnistyä jossain harvinaisessa tilanteessa uudelleen, ja niin sanottujen ohjelmiston vahtikoira-ajastimien käyttö on mahdollista, edellä kuvattu toiminta kääntyy eduksi. Vahtikoira-ajastin toimii siten, että se odottaa saavansa herätteen ennalta määrätyin aikavälein tai useammin. Jos aikaväli ylittyy, vahtikoira käynnistää ohjelman uudelleen. Kun siis herätteiden antaminen jätetään alimman prioriteetin tehtävän vastuulle, voidaan taata, että ohjelma ei voi koskaan jäädä loputtomaan silmukkaan ilman, että se palautuisi vahtikoiran avulla toimintakuntoon. Jos nimittäin loputtomaan silmukkaan jääminen tapahtuu tätä tehtävää ylemmän prioriteetin tehtävässä, alimman prioriteetin tehtävä ei saa koskaan suoritusaikaa eikä siten voi antaa herätteitä vahtikoiralle. Kun lisäksi herätteen antaminen sijoitetaan tämän tehtävän suorituksen alkuun, aloitettiin tehtävä sitten tasaisin väliajoin tai annettiin sen kiertää loputtomassa pääsilmukassa, taataan se, että jos tehtävä jää kiertämään tätä sisempään silmukkaan, ei herätteen antamisen ohjelmariviä enää saavuteta, ja taas lopputuloksena on uudelleen käynnistys.

Vahtikoiramenetelmässä joudutaan luonnollisesti olettamaan, ettei yksikään tehtävä joudu koskaan suorittamaan vahtikoiralle valittua aikaväliä pidempiä osatehtäviä. Jos näin on, joudutaan herätteiden antaminen suorittamaan poikkeuksellisesti tämän osatehtävän sisällä, ja tällöin on ohjelmoijan vastuulla olla huolellinen sen suhteen, että tämä silmukka ei voi jäädä

loputtomaan suoritukseen. Suositeltavaa on merkitä järjestelmäkellon aika muistiin ennen riskialttiiseen osaan siirtymistä, ja sen jälkeen verrata nykyhetkeä muistiin merkittyyn aikaan jatkuvasti tämän osatehtävän aikana. Jos aikaeron havaitaan kasvaneen sopivaa ylärajaa suuremmaksi, käynnistetään ohjelma uudelleen.

Myös laitteisto tuo mukanaan tiettyjä haasteita reaaliaikajärjestelmän suunnitteluun. Ohjelmisto mittaa kellonaikaa laitteistossa olevan kiteen avulla. Kide saattaa kuitenkin värähdellä tarkoitettua nopeammin tai hitaammin riippuen muun muassa sen lämpötilasta ja laitteiston käyttöjännitteestä [11, s. 49]. Jos kaksi laitetta vaihtavat viestejä keskenään ja molemmissa on oma kiteensä, on huomioitava mahdollisuus, että toisen kellonaika edistää ja toisen jätättää. Mikäli laite vertaa viestissä olevaa kellonaikaa omaan kellonaikaansa, saatetaan viestiä virheellisesti pitää vanhentuneena. Tämä voidaan ratkaista sopimalla jokin viitekellonaika, jonka kukin laite voi tarkistaa ja korjata sen perusteella omaa kellonaikaansa. Viitekellonaika voidaan saada järjestelmän ulkopuolelta, kuten NTP-protokollaa tukevalta palvelimelta, tai se voi olla tietyn järjestelmään kuuluvan laitteen kellonaika.

Aina oikein toimivan viestinvälityksen suunnittelu sisältää oman ongelmakenttänsä. Jos viestin saapumisajalla on takaraja, on toiminta rajatapauksissa aina mietittävä tarkasti. Toinen siirtosuunta saattaa esimerkiksi olla hieman hitaampi kuin toinen, jolloin toinen laitteista ottaa jatkuvasti viestit vastaan ja toinen hylkää ne aina. Huolimaton viestinvälityksen suunnittelu voi johtaa siihen, että kun tällainen vika todetaan, ei jää muita vaihtoehtoja kuin jo rakennetun järjestelmän uudelleen suunnitteleminen. Eräs ominaisuus, joka tekee järjestelmästä vähemmän alttiin viestinvälitykseen liittyville ongelmille, on viestien *idempotenssi*. Se tarkoittaa, että jos yksi viesti saapuu laitteelle useaan kertaan, se käsitellään samoin kuin, jos se olisi saapunut vain kerran [11, s. 110].

3.2 Puskurin ylivuotovirheet

Puskurin ylivuotovirheeseen perustuu merkittävä osa tunkeutujien hyväkseen käyttämistä haavoittuvuuksista. Tilastojen mukaan vuosina 2001–2004 keskimäärin noin 22 % kaikista ohjelmistojen haavoittuvuuksista kuului tähän joukkoon [5, s. 10]. Virhe tapahtuu, kun ohjelmistossa olevaan yhtenäiseen tietoalueeseen eli puskuriin yritetään kopioida enemmän tavuja kuin sinne mahtuu. Tällöin ylimääräiset tavut tulevat kopioiduksi puskurin ulkopuolelle, jossa on jo muuta tietoa tai konekielisiä komentoja. Tavallisesti tästä seuraa ainoastaan se, että ohjelma suorittaa käyttöjärjestelmän näkö-

kulmasta virheellisen toiminnon, ja käyttöjärjestelmä turvallisuussyistä lopettaa sen.

Tunkeutujan on kuitenkin mahdollista käyttää virhettä hyväkseen, ja sijoittaa puskurin yli menevään osaan konekielisiä komentoja ja muuta ohjaustietoa. Yleensä tunkeutuja sijoittaa sinne hyppykäskyn, jolla suoritus saadaan siirtymään tunkeutujan haluamaan ohjelmakoodiin. Se on saatettu antaa mukana syötteessä, tai sitten se on jokin järjestelmässä jo oleva ohjelmakoodi, jonka suorittamisesta on tunkeutujalle hyötyä. Jos tunkeutuminen tehdään taitavasti, lopussa on vielä hyppykäsky takaisin alkupisteeseen, jolloin käyttäjän on lähes mahdotonta havaita, että jotain epätavallista on tapahtunut.

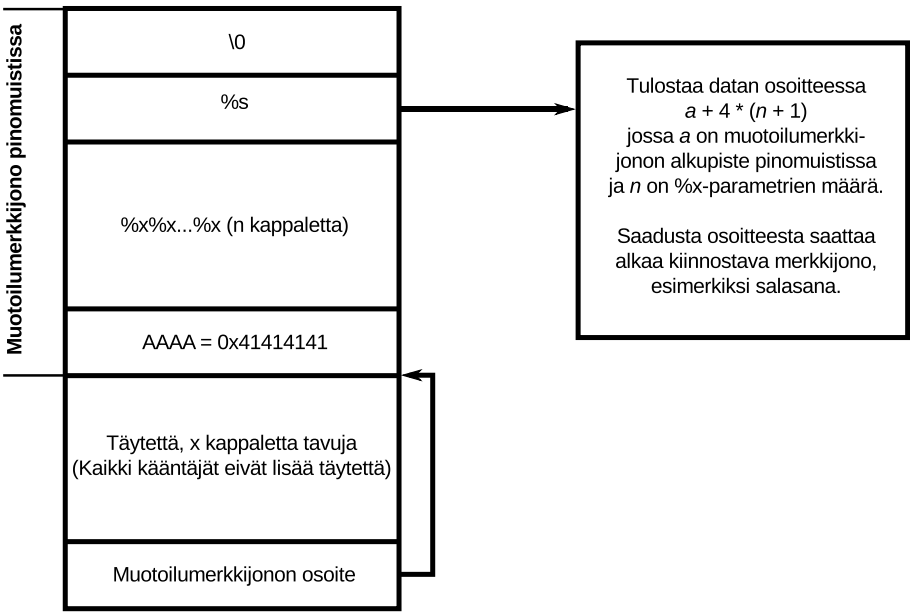
Puskurin ylivuotovirheet jaetaan kahteen ryhmään. Niihin, joissa virhe tapahtuu pinomuistissa olevassa puskurissa ja niihin, joissa kyseessä on kekomuisti. Jos kuitenkin kekomuistissa on erilliset data- ja kontrollialueet, ei tämän tyyppistä puskurin ylivuotovirhettä voida käyttää järjestelmään tunkeutumiseen [5, s. 162].

Nämä virheet voidaan välttää tarkistamalla aina kopioitaessa, että kopioitava osa mahtuu puskuriin. C-ohjelmointikielen strcpy-käsky on yleinen virheiden aiheuttaja. Tämän vuoksi on olemassa strncpy-käsky, jolle voidaan lisäksi antaa parametrina puskurin pituus, ja tällöin vältytään aina kopioimasta sen ulkopuolelle. Tämän käskyn käyttäminen on kuitenkin vaivalloisempaa, koska hyvän ohjelmointitavan noudattamiseksi on puskurin pituudesta tehtävä ohjelmaan oma vakionsa.

3.3 Muotoilumerkkijonoon perustuva hyökkäys

Ohjelmistoissa on useimmiten käyttöliittymä, johon annetaan syötteitä. Jos tällainen syöte annetaan sellaisenaan jollekin funktiolle, joka käsittelee sitä muotoilumerkkijonona, on tunkeutujan mahdollista tutkia laitteen muistin sisältöä. Tämä tapahtuu antamalla merkkijonossa argumentteja, joita todellisuudessa ei ole ohjelmakoodissa yhdistetty mihinkään muuttujaan. Tällöin argumentit saavat arvonsa siitä kohdasta muistia, jossa tämän muuttujan olisi pitänyt olla.

Hyökkäys vaatii hieman valmistelua, koska hyökkääjä ei aluksi tiedä, tulisiko muotoilumerkkijonon jälkeen nähtävät tavut tulkita merkkijonona vaiko lukuina. Tämä selvitetään kokeilemalla eri vaihtoehtoja. Koska kuitenkin merkkijonoja voi olla mahdollista selvittää pinomuistista, korostuu sen tärkeys, että oikeat salasanat eivät esiinny edes muistissa selväkielisinä vaan ne tarkistetaan aina esimerkiksi vertaamalla salasanojen tarkistussummia.



Kuva 3.2: *Esimerkki muotoilumerkkijonon syöttämiseen perustuvasta hyökkäyksestä [5, s. 289]*

Kuvassa 3.2 on esitetty muotoilumerkkijonoon perustuva hyökkäys, kun käytetään merkkijonoa ”AAAA%x%x...%x%s” ja %x-osien oikea määrä on ensin selvitetty kokeilemalla käyttäen lähes samanlaista merkkijonoa. Merkkijono alkaa kirjaimilla *AAAA* siksi, että näin saadaan selville kääntäjän mahdollisesti lisäämän täytteen määrä, kuvan mukaisella tavalla. Olettaen että muistiavaruus on 32-bittinen, AAAA on muistissa varastoituna kuin heksadesimaaliluku 0x41414141.

Kun yksi muotoilumerkkijonon %x-parametreista antaa tulosteeseen luvun 0x41414141, voidaan muotoilumerkkijonon alkupiste päätellä tämän parametrin järjestysnumerosta. Sen jälkeen tarvitsee ainoastaan lisätä niin monta %x-parametria, kuin halutaan edetä neljän muistiosoitteen askeleita, päätyen lopulta etsittävään merkkijonoon pinossa. Se voi olla esimerkiksi oikea salasana, jota vastaan käyttäjän syöte tarkistetaan. Jos ympäristö onkin vaikkapa 16-bittinen, käytetään kahden merkin mittaista merkkijonon alkua ja huomioidaan, että kukin %x-parametri vastaa kahta tavua.

Jos käytetty kääntäjä tukee %n-parametria, on jopa mahdollista muuttaa muistin sisältöä. Menetelmän periaate on luettavissa viitteestä [5, s. 291]. Tämän työn kannalta ei tällä ole merkitystä, koska DynamicC ei %n-parametria tue.

4. LÄHTÖKOHDAT

Työtä aloitettaessa Telcont Oy oli määritellyt uuden ilmoituksensiirtoprotokollan, jota toteutettava laite tulisi käyttämään. Protokollan nimi on *ACProto*, sanoista Alarm Control Protocol ja se on kuvattuna tarkemmin aliluvussa 4.4. Samaten käytettäväksi laiteympäristöksi oli päätetty Rabbit Semiconductorin valmistama RCM3700-moduuli. Telcont Oy oli suunnitellut sille pohjakortin, josta moduuli saisi virtansa ja josta hälytystulot ja relelähdöt liitettäisiin sen suorittimen rinnakkaisporttien bitteihin.

Ohjelmistosuunnittelun vesiputousmallin käyttäminen ei ollut projektissa mahdollista. Työtä aloittaessa oli päätetty vain kaksi tavoitetta: Saada laite välittämään hälytys Computec Oy:n valmistamaan valvomolaitteeseen keskitinlaitteen kautta, ja toteuttaa selaimella käytettävä käyttöliittymä josta hälytyspisteen tietoja voitaisiin asettaa. Sitten etsittäisiin kyseistä valvomoa käyttäviä asiakkaita ja jatkettaisiin kehitystä heidän tarpeidensa ja toiveidensa mukaisesti. Asiakaslähtöinen *ketterä* toimintamalli soveltui lähtökohtiin paremmin.

Tärkeä osa ketteriä menetelmiä on ohjelmakoodin jatkuva refaktorointi, jonka tavoitteena on estää huonon suunnittelun kasautuminen projektissa. Refaktoroinnissa tietty osa ohjelmakoodia kirjoitetaan uudelleen, tekemään sama asia kuin aiemmin, mutta paremmin. Jos suunnittelu- ja toteutusvirheitä joudutaan kiertämään toisessa osassa koodia joka käyttää edellistä, ja sitten taas kiertämisen tuottamia ongelmia kolmannessa ja niin edelleen, tulee ohjelma erittäin vaikeaksi hallita ja toisaalta sen refaktorointi olisi valtava projekti, kun kaikki osat olisi refaktoroitava. Tavoitteena on yksinkertaistaa ohjelmakoodin osia ja yhdistää niitä uudelleen käytettäväksi koodiksi sekä poistaa redundanssia. [17, s. 93]

Ketterissä käytännöissä suositellaan myös pariohjelmointia, jossa kaksi ohjelmoijaa työskentelee yhdessä yhdellä koneella, toisen ohjelmoidessa ja toisen ajatellessa prosessia eteenpäin, ratkaisten valmiiksi ohjelmoijan eteen tulevia ongelmia. Tämä ei ollut sellaisenaan mahdollista, koska työntekijöitä oli siinä vaiheessa tekijän lisäksi vain yksi, ja hän oli vastuussa ATS:n kehittämisestä. Yhteistyötä kuitenkin saatiin tehtyä tehokkaasti, koska ongelmat olivat usein samoja, sillä kyseessä oli saman protokollan asiakas- ja palvelin-

laite. Kun oli ratkaissut ongelman omassa projektissaan oli mahdollista kuvailla se toiselle, ennen kuin se tuli hänen projektissaan eteen. Yhteistyötä helpotti samassa tilassa työskentely ja näköyhteys näyttöjen välillä.

4.1 Laite- ja kehitysympäristö

Laitealustaksi valittu *RCM3700*-moduuli sekä sen kehitysympäristö on amerikkalaisen yrityksen, Rabbit Semiconductorin valmistama. Yrityksen on omistanut vuodesta 2006 lähtien Digi International Inc., kun taas aiempi omistaja oli Z-World, Inc. Alusta valittiin toisaalta moduulin edullisuuden vuoksi ja toisaalta siksi, että kehitysympäristö sisältää suuren määrän projektissa välttämättömiä ohjelmakirjastoja, kuten *MicroC/OS-II*-reaaliaikaympäristö sekä *PPP*-kirjasto *HTML*-sisällön välittämiseen sarjaportin avulla. Ohjelmakirjastojen lisenssi oli varsin salliva. Kirjastoihin sai tehdä muutoksia kysymättä lupaa lisenssinhaltijalta ja niiden käyttäminen oli maksutonta.

4.1.1 Dynamic C

Laitealustaksi valittu, Rabbit Semiconductorin kehittämää moduulia ohjelmoidaan valmistajan tarjoamalla kehitysympäristöllä nimeltä *Dynamic C*. Se käyttää omaa murrettaan C-ohjelmointikielestä, joka poikkeaa merkittävästi ANSI C -standardista. Siksi kynnys käyttää yleisesti saatavilla olevia ohjelmakirjastoja on korkea, koska niiden muokkaaminen Dynamic C:n vaatimaan muotoon on vaivalloista. Onnellisena sivuvaikutuksena Dynamic C ei tue *printf* -funktion muotoilumerkkijonon elementtiä *%n* [3, s. 362], mikä tekee yhden aliluvussa 3.3 mainituista hyökkäysmenetelmistä mahdottomaksi.

Seuraavalla sivulla olevasta ohjelmalistauksesta 4.1 ilmenee Dynamic C:n erikoinen tapa esitellä funktiot ohjelmakirjastoissa. Kieli ei tue otsikkotiedostoja, vaan esittely tapahtuu ohjelmakooditiedostossa välittömästi ennen funktion määrittelyä. Kommenttilohkolla, joka alkaa ”/***” välitetään komentoja kääntäjän esikäsittelijälle, ja muun muassa ilmaistaan, että kyseessä on funktion esittely, käskyllä *BeginHeader*.

DynamicC:ssä ohjelmakirjaston lähdekooditiedoston pääte ei ole tavanomainen ”.c”, vaan ympäristö vaatii päätettä ”.lib”. Tällaiset poikkeamat tekevät kolmansien osapuolien tarjoamien työkalujen käyttämisestä vaivalloista. Esimerkiksi projektissa ohjelmakoodin kirjoittamiseen käytetty *XEmacs*-editori ei tunnista kirjastojen sisältävän C-ohjelmointikieltä ellei sitä erikseen määrittele tunnistamaan lib-päätettä C-ohjelmakooditiedostoksi. Myös projektin dokumentointiin käytetty *Doxygen*-ohjelma, josta kerrotaan lisää

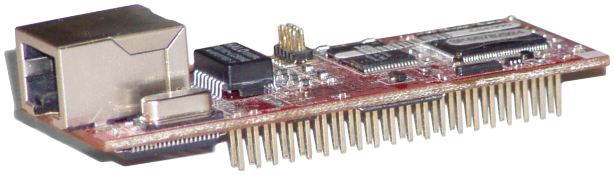
```
/*** BeginHeader crypt_message */
char crypt_message(uint8* target_data, uint8* source_data, uint16 len,
                   uint8 direction);
/*** EndHeader */

/** Encrypts and decrypts ACProto data.
 *
 * Target pointer and source pointer are allowed to be the same,
 * in which case the processed data overwrites the original
 *
 * @param target_data   Pointer to where we place the processed data
 * @param source_data   Pointer to the original data
 * @param len           Length of the original data
 * @param direction     DIRECTION_CRYPT for encryption, or
 *                      DIRECTION_DECRYPT for decryption
 *
 * @return
 *      0 if successful,
 *      1 if there was an error, or
 *      AGAIN if another task was using the function
 */
char crypt_message(uint8* target_data, uint8* source_data, uint16 len,
                   uint8 direction)
{
```

Ohjelma 4.1: Näyte Dynamic C:n tavasta esitellä funktiot sekä Doxygen-dokumentaatiosta

aliluvussa 4.1.4, tarvitsee ohjelmakoodiin ylimääräistä ohjausta, koska se tulkitsee väärin tiettyjä Dynamic C:n käytäntöjä, kuten kekomuuttujien alustamisen #GLOBAL_INIT {...} -ohjelmakoodilohkolla. Ohjelmalistauksesta 4.1 ilmenee myös Doxygenin käyttämä syntaksi ohjelmakirjaston funktioiden dokumentointiin. Doxygenille välitetään tietoja kommenttilohkolla, joka alkaa ”/**”.

4.1.2 MicroC/OS-II-reaaliaikaympäristö

Dynamic C tukee ohjelmien rinnakkaisuutta kahdella tavalla. Kääntäjä tukee yhteistyömoniajoa (englanniksi *cooperative multitasking*) ilman erillistä kirjastoa. Yhteistyömoniajon käyttäminen on joissakin projekteissa perusteltua sen vuoksi, että se jättäisi enemmän muistia vapaaksi kuin erillisen kirjaston vaativa, irroittava reaaliaikaympäristö. Toisaalta ohjelmiston toimintavarmuutta on vaikeampaa taata yhteistyömoniajoa käyttäen. Siinä kaikki tehtävät ovat tasa-arvoisessa asemassa keskenään, ja toinen tehtävä pääsee suoritukseen vain, kun suorituksessa parhaillaan oleva tehtävä oma-aloitteisesti lopettaa suorituksen. Näin ollen olisi mahdollista, että jokin kriittinen tapahtuma, kuten hälytyksen havaitseminen silmukassa, jäisi



Kuva 4.1: *RCM3700 -moduuli*

odottamaan toisarvoista tapahtumaa, kuten laiteasetusten näyttämistä etäkäyttäjälle. Tämä ei ole laitteen käyttötarkoituksen huomioon ottaen hyväksyttävää.

Toisaalta kehitysympäristön mukana toimitetaan kolmannen osapuolen laatima, Dynamic C:n erikoista syntaksia varten muokattu irrottava reaaliaikaympäristö, *MicroC/OS-II*. Tällaisten reaaliaikaympäristöjen toimintaperiaate ilmenee aliluvusta 3.1. MicroC/OS-II on sertifioitu käytettäväksi kriittisissä, eli *safety critical* -luokkaan kuuluvissa järjestelmissä [12, s. xv]. Tämä kirjasto on jaettu osiin siten, että on mahdollista säästää kallisarvoista muistia ottamalla käyttöön vain ohjelmistossa tarvitut osat.

Tehtävienhallinnan lisäksi MicroC/OS-II tarjoaa muitakin reaaliaikaympäristöille tyypillisiä palveluja, kuten viestijonot ja postilaatikot. Nämä ovat menetelmiä tehtävien väliseen viestintään. Samaten käytössä on semaforit, mutexit ja ehtomuuttujat, joilla voidaan estää tehtäviä käyttämästä yhtäaikaa sellaisia resursseja, jotka voisivat aiheuttaa yhtäaikaa käytettynä ohjelmavirheen. Näistä palveluista voi lukea lisää mistä tahansa reaaliaikajärjestelmiä käsittelevästä kirjasta, kuten lähteestä [6]. Myöhemmin käsitellään vain viestijonoa, koska se on ainut tämän laitteen käyttämä MicroC/OS-II:n palvelu.

4.1.3 RCM3700

ATM0602-laite toteutettiin käyttäen RCM3700-moduulia, joka on esitettynä kuvassa 4.1. Se edustaa Rabbit Semiconductorin tuotelinjan hinnottelun keskitasoa. Kaikki RCM3000-sarjan moduulit sisältävät *Rabbit 3000* -suorittimen ja poikkeavat ainoastaan moduulissa olevan muistin määrän suhteen. Käytetyssä moduulissa on 512 kilotavua flash-muistia ohjelmakoodia varten, sekä saman verran SRAM-muistia. Lisäksi moduulissa on erillinen *serial flash* -piiri, jota käytetään suorittimen B-sarjaportin avulla laitteen massamuistina.

Flash-muistin käyttö on hankalampaa kuin RAM-muistin. Flash-muistin muistiavaruus jakautuu lohkoihin, ja sen sisältöä pyyhittäessä on pyyhittä-

vä aina koko lohko. Jos siis aikoo kirjoittaa uudelleen tietyn alueen muistia, kuten vaikkapa merkkijonon, joka osuu kahden eri lohkon alueelle, on ensin luettava molempien lohkojen sisältö RAM-muistiin, tehtävä muutos siellä, pyyhittävä lohkot ja sitten kirjoitettava molemmat lohkot kokonaisuudessaan takaisin flash-muistille.

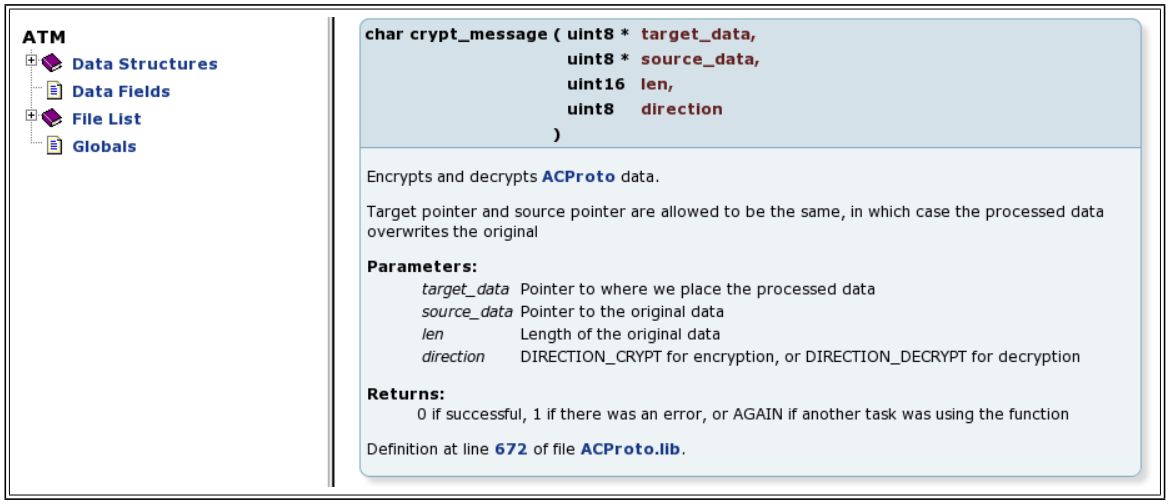
Telcont oli aiemmin toteuttanut toista tuotetta varten ohjelmakirjaston, joka tekee tämän automaattisesti ja jota käytettiin tässä projektissa uudelleen. Usein flash-muistia varten toteutetaan tiedostojärjestelmä, joka pyrkii hajauttamaan käytön tasaisesti kaikkiin lohkoihin, koska valmistaja takaa kullekin lohkolle vain tietyn määrän pyyhkimiskertoja ennen kuin siitä tulee käyttökelvoton. ATM0602-laitteen kuitenkin arvioitiin tarvitsevan flash-muistin sisällön muuttamista niin harvakseltaan, että tätä pidettiin tarpeettomana.

RCM3700-moduulin suoritin perustuu kahdeksanbittiseen arkkitehtuuriin. Se muistuttaa Z80-, Z180- ja HD64180-suorittimia ja sen suurin tuettu kellotaajuus on 55,5 MHz. Arkkitehtuuria on muokattu edellä mainituista, vanhemmista suorittimista nimenomaan C-ohjelmointikielen tarpeita silmällä pitäen. [1, s. 1]

4.1.4 Doxygen

Projektin alusta asti oli tiedossa, että perinteistä vesiputousmallia ei olisi järkevää pyrkiä noudattamaan, koska asiakasvaatimukset eivät olleet perustoimintoja lukuunottamatta tiedossa. Oli selvää, että suunnitelmissa tapahtuisi äkillisiä muutoksia sen mukaan, mitä asiakastoiveita ensimmäiset myydyt tuotteet toisivat tullessaan.

Ohjelmisto nimeltä Doxygen soveltui näihin lähtökohtiin erinomaisesti. Se muodostaa projektin dokumentaation automaattisesti lähdekoodissa annettujen ohjauskomentojen mukaisesti. Doxygenilla voidaan tuottaa muun muassa Adoben kehittämässä PDF-muodossa olevia dokumentteja, HTML-muotoista hypertekstiä tai yksinkertaista, mutta yhteensopivaa RTF-tekstiä. Ohjelmalle tarvitsee vain kertoa, mitä erilaisia tiedostoja halutaan, ja se tuottaa ne.



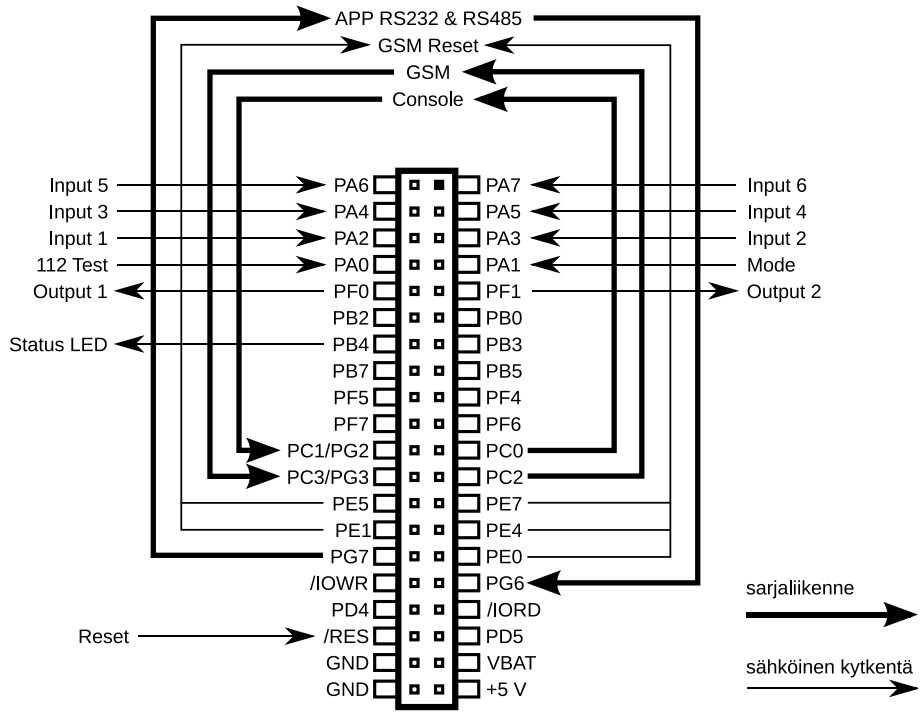
Kuva 4.2: *Esimerkki Doxygen-ohjelman tuottamasta dokumentaatiosta*

Kuvassa 4.2 näkyy hypertekstinä se osa dokumentaatiosta, jota vastaava lähdekoodi esitettiin ohjelmalistauksessa 4.1. Huomionarvoista kuvassa on myös vasemmalla oleva navigaatiopalkki, josta dokumentaation eri osia voidaan selata. Suositeltavaa Doxygeniä käytettäessä on, että lähdekoodissa dokumentoidaan vähintään jokaisen funktion parametrit ja paluuarvot, sekä itse määriteltyjen datatyyppien kenttien tarkoitukset. Funktioiden yhteyteen kirjoitetaan lyhyt kuvaus niiden käyttötarkoituksesta, mahdollisista esiehdoista funktion kutsumiselle ja muusta huomionarvoisesta. Samaten jokaisen ohjelmakooditiedoston alkuun kirjoitetaan kuvaus tiedostosta yleisesti, ja tiedostojen muodostamista ohjelmiston moduuleista kirjoitetaan myös kuvaukset.

Menetelmän etuna on se, että sillä vältetään yleinen vaara dokumentaation päivittämisen unohtumisesta, kun työt alkavat kasaantua. Kun dokumentaation sisältämät tiedot ovat lähdekooditiedostossa, niiden päivittäminen vie huomattavasti vähemmän ohjelmoijan aikaa. Tuloksena on se, että projektin muut ohjelmoijat voivat luottaa dokumentaation ajantasalla olemiseen silloin, kun sitä eniten tarvitaan, eli kun työmäärä on suurin, ja vapaat resurssit perinteisemmän dokumentaation päivittämiseen siten vähimmillään.

4.2 Kytkennät ja rajapinnat ATM0602 -moduulissa

RCM3700-sarjan moduuleihin kuuluvien suorittimien sarja- ja rinnakkaisportteja käytetään moduulin pohjassa olevan piikkiriman avulla. Moduulin ostaja suunnittelee pohjakortin, jolla piikit ohjataan sähköisesti oikeisiin

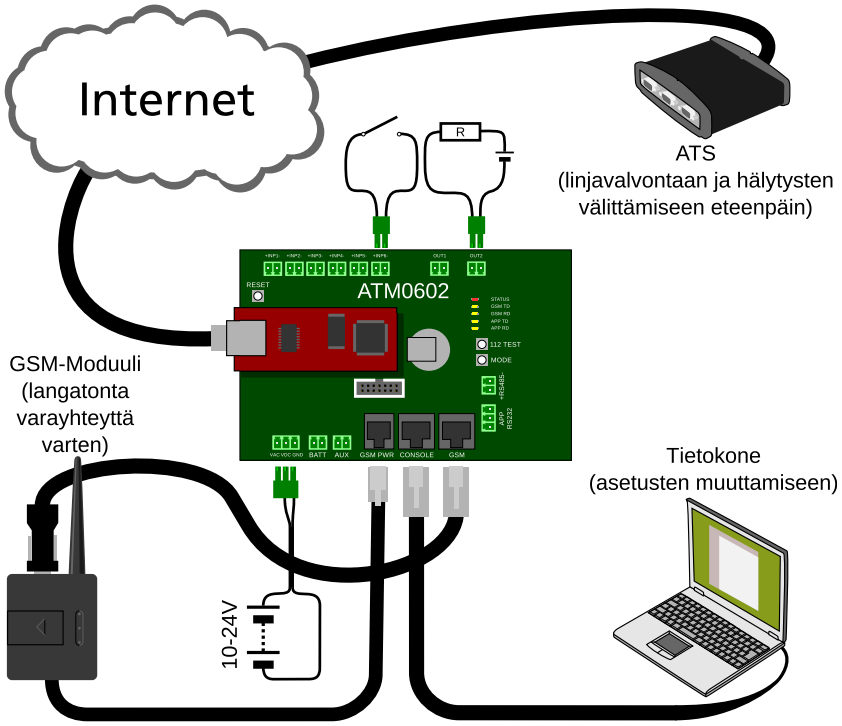


Kuva 4.3: *RCM3700 -moduulin sähköiset rajapinnat ja niiden käyttö ATM0602:ssa*

paikkoihin. Kuvassa 4.3 on nähtävillä moduulin tarjoamat rajapinnat sekä niiden käyttö Telcontin pohjakortissa ja se on mukailtu lähteen [2, s. 24] kuvasta.

Kuvassa *Input 1–6* tarkoittaa tuotteen tarjoamia kuutta hälytyssilmukkaa, ja *Output 1–2* tarkoittaa sen kahta etäohjattavaa relettä. *Mode* ja *112 Test* ovat painikkeita, joista ensiksi mainitulla avataan asetusten muuttamiseen tarkoitettu käyttöliittymä ja jälkimmäisellä testataan, että GSM-signalointiin perustuva varayhteys toimii. Painikkeen nimi johtuu siitä, että testaus kuuluu pakollisena osana asennusmenettelyyn, kun hälytyksiä halutaan välittää hätäkeskuslaitokselle. *Status LED* antaa tietoa laitteen tilasta, kuten internet-yhteyden toimimisesta ATS:lle, ja kaksi *APP*-porttia ovat sarjaliikenneportteja mahdollisia lisälaitteita varten. *GSM*-sarjaportti on varayhteyden mahdollistavaa GSM-moduulia varten, kun taas *Console*-sarjaportin avulla kytketään tietokone kiinni laitteeseen käyttöliittymää varten.

Seuraavalla sivulla olevassa kuvassa 4.4 esitetään pohjakortin tarjoamat rajapinnat havainnollisemmin. Siinä näkyy myös *reset*-painike, jota ei ollut edellisessä kuvassa siksi, että se katkaisee moduulin virransyötön sen sijaan, että se olisi liitetty suorittimen reset-sisäänmenoon. Näin turvataan laitteen toimintakuntoon saaminen, jos pahin tilanne tapahtuisi ja suorittimen

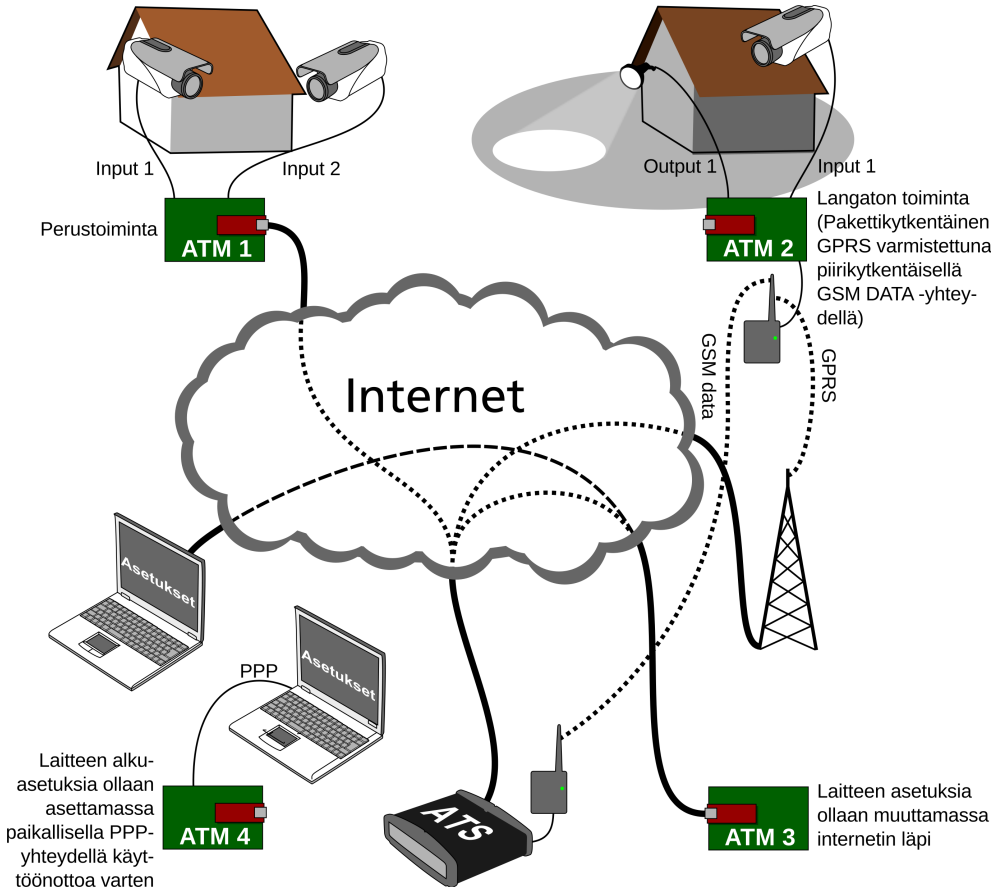


Kuva 4.4: *Kaaviokuva ATM0602-moduulista ja sen tärkeimmistä liitynnöistä*

suunnitteluvirheen vuoksi se ei vastaisi mihinkään komentoihin. ATM0602-laitteita asennetaan usein sellaisiin paikkoihin, joissa virransyötön sekä akusta saatavan varajännitteen irroittaminen on erittäin vaivalloista.

Laitteen kahdella muulla painikkeella on myös toinen käyttötarkoitus: kun ne painetaan yhtäaikaa pohjaan ja tämän jälkeen vielä reset-painike, pohjaan ja takaisin ylös, laite palauttaa tehdasasetukset. Laitteen käyttöliittymä on suojattu asiakkaan valitsemalla salasanalla, joten jos se unohtuu, saadaan laite jälleen käyttökuntoon tällä tavalla. Samalla salasana pyyhkiytyy kokonaan muistista, joten tällä tavalla ei luvaton käyttäjä saisi sitä selville.

Kuvassa näkyvät kaksi releisiin kytkettyä pistoketta ovat kuorman, kuten sähkölukon tai valaisimen kytkemistä varten. Käsky ohjata relettä välitetään laitteelle ATS:n kautta, jolloin se kulkeutuu salattuna sekä varmistettuna Telcontin patentoimalla [7] tietoturvamenetelmällä. Jatkokehitystä ajatellen olisi mahdollista sallia asiakkaalle automaattisia toimenpiteitä, jotka ATS toteuttaisi valittuina aikoina. Näin voitaisiin vaikkapa lukita kerrostalon ovet iltaisin ja avata lukitus aamuisin.



Kuva 4.5: *Laitteen käyttötapoja ilmoituksensiirtoon ja asetusten muuttamiseen*

4.3 Moduulin toiminta osana hälytyksensiirtojärjestelmää

Laitteen toimintaa osana sen ja ATS:n muodostamaa hälytyksensiirtojärjestelmää on havainnollistettu kuvassa 4.5. Ensimmäinen kuvatuista laitteista on liitetty ilman varayhteyttä ja käytössä on kaksi hälytyssilmukkaa.

Kuvassa esitetyt valvontakamerat sisältävät liiketunnistimen, joka avaa releen. Näin sähköinen silmukka katkeaa, jolloin ATM0602 välittää hälytystiedon ATS:lle. ATM0602:n kannalta ei ole merkitystä, mitä ATS hälytyksen suhteen tekee. Kun ATM0602 on saanut vahvistuksen hälytystiedon saapumisesta ATS:lle, siirtyy vastuu hälytyksen välittämisestä sille. Se saattaa esimerkiksi siirtää sen Alerta- tai TurvaLinja-yhteyttä pitkin vartiointiliikkeen valvomoon, mutta ATM0602:n toiminnan kannalta tällä ei ole merkitystä.

Toinen kuvassa esitetyistä laitteista sisältää varayhteyden tarjoavan GSM-moduulin. Se on asetettu langattomaan toimintaan, jolloin pääyhteytenä toimii GPRS-yhteys internetiin ja varayhteytenä GSM. ATS:n kannalta GPRS ja kiinteä internet-yhteys eivät eroa muilta kuin siltä osin, että GPRS:n tapauksessa oletetaan datapakettien liikkumiseen liittyvät viiveet

kaksinkertaisiksi. Jos ATM0602 ei saa ATS:ltä yhteydenottoa käyttöliittymän kautta määritellyin välein, se soittaa tavanomaisen GSM-äänipuhelun ATS:än liitettyyn GSM-moduuliin. Puhelujen välillä se pyrkii muodostamaan jälleen GPRS-yhteyttä ATS:än. Jos varayhteyden varassa ollessaan ATM0602:n hälytyssilmukka aktivoituu, se soittaa GSM-datapuhelun ATS:lle ja välittää hälytystiedon tällä tavalla. GSM-datapuhelu on äänipuhelua muistuttava, piirikytkentäinen tiedonsiirtomenetelmä. Yhteyden siirtonopeus on tässä tapauksessa 9 600 bittiä sekunnissa. Hälytykseen liittyvän datan määrä on vähäinen ja sen siirtämiseen kuluu noin yksi sekunti. Teleoperaattorit laskevat puhelujen kestoa tätä pidemmillä askeleilla, joten puhelu maksaa yhden laskentayksikön verran.

Kuvan 4.5 kolmannen ja neljännen laitteen käyttöliittymä on aktiivisena. Kolmas laite on samaan aikaan valmiina siirtämään hälytyksiä, kun sen käyttöliittymää käytetään internetin läpi. Irrottavan reaaliaikaympäristön valitseminen mahdollistaa sen, että laiteessa hälytyssilmukoiden tarkkailu on korkeammalla prioriteetilla kuin käyttöliittymän palveleminen, eikä hälytysten perillemeno näin vaarannu etäkäytöstä.

Neljäs laite ei ole vielä kytketty internetiin, vaan sille ollaan syöttämässä alkuasetuksia sarjakaapeliyhteyden avulla, jotta yhteys ATS:ään olisi mahdollinen. Tämän jälkeen laitetta voidaan käyttää etänä, eikä suoraa yhteyttä enää tarvita. Tästä on tosin poikkeuksena yhteysasetukset. Ne on aina asetettava suoralla yhteydellä, koska, jos etäyhteyttä käytettäessä tehtäisiin virhe ja laite vaihtaisi kelvottomiin yhteysasetuksiin, ei etäyhteyskään olisi enää mahdollinen, eikä tilannetta saataisi korjattua kuin menemällä paikalle. Tänä aikana hälytyksetkään eivät kulkisi väärien asetusten vuoksi.

4.4 ACProto-hälytyksensiirtoprotokolla

Telcont Oy:n tavoitteelle uudistaa ilmoituksensiirto Suomessa asetti rajoituksia aliluvussa 2.4 kuvattu, 70-luvulta peräisin oleva tekniikka, jonka kanssa uuden järjestelmän olisi oltava yhteensopiva. Juuri ennenkuin tässä diplomityössä kuvatun laitteen kehittäminen alkoi, oli yrityksessä määritelty uusi ilmoituksensiirtoprotokolla jota laite käyttäisi, nimeltä *ACProto*, sanoista Alarm Control Protocol. Koska laitteessa olisi useita hälytystuloja ja relelähtöjä, oli ensin päätettävä, miten pisteosoitteet jaettaisiin niiden välillä. Laitteiden asentaminen olisi epäkäytännöllistä, jos asentajien olisi varmistettava toisiltaan, etteivät he ole antaneet silmukoille samaa osoitetta.

ComNetin määrittelemään osoiteavaruuteen kuuluvan laitteen valmistajan on päätettävä tapa, jolla osoiteavaruus jaetaan kunkin samassa ylä- ja alaverkossa olevan laitteen kesken niin, ettei yksikään pisteosoite kuu-

lu useammalle laitteelle. Tätä varten määriteltiin ACProtossa käsite *Terminal Address*, pääteosoite. ComNetin määrittelemä osoiteavaruus ei sisällä tätä käsitettä, vaan se kuuluu ACProtoon. Päädyttiin siihen, että laitteiden pääteosoitteet olisivat välillä 5–250 ja pääteosoitteesta saataisiin ACProtossa määritellyllä laskukaavalla pisteosoite sen kullekin tulolle ja lähdölle.

Asentajien olisi ainoastaan varmistuttava, ettei samaa pääteosoitetta annettaisi useammalle laitteelle. Koska kuhunkin alaverkkoon kuuluu 65535 pisteosoitetta jaettavaksi laitteiden kesken, saadaan tästä kullekin laitteelle oma kahdensadan pisteen osoiteavaruus, kun laitteita ei voi olla enempää kuin 245. Laitteen osoiteavaruus voisi toki olla hieman suurempikin, mutta laskeminen helpottuu, kun luku on 200.

Laskukaava on siis pääteosoite kertaa 200. Tällä tavalla eri laitteiden osoiteavaruuksien ensimmäiset pisteosoitteet ovat kahdensadan osoitteen välein toisistaan ja kullekin laitteelle jää siten 200 pistettä omaan käyttöönsä. Käyttämällä näin suurta osoiteavaruutta mahdollistetaan se, että yhteyskäytäntöä tukeviin laitteisiin voitaisiin myöhemmin liittää enemmän pisteitä tarjoava lisälaite, kuten *Modbus*-isäntälaite. Laitteella itselläkin on oma pisteosoitteensa, joka on sama kuin pääteosoite. Näin voidaan ilmaista esimerkiksi laitevika, joka koskee kaikkia laitteeseen liitettyjä silmukoita. Samaten mistä tahansa pisteosoitteesta voidaan päätellä, mihin laitteeseen se kuuluu jakamalla osoite kahdellasadalla ja ottamalla tuloksesta kokonaislukuosa.

Protokolla oli suunniteltu yhteensopivaksi Computec Oy:n Comsystemin kanssa, mitä tietosisältöön tulee. Comsystemiin kuuluu esimerkiksi hälytyslaji, joka on luku väliltä 1–255. Eräs yleisesti käytetyistä hälytyslajeista ja ATM0602:n oletuslaji, on ensimmäisen kiireellisyysluokan hälytys, jota vastaa luku 2. ACProtossa käytetään samoja numeroja tarkoittamaan samoja hälytyslajeja kuin Comsystemissä.

Poikkeuksen muodostaa laji 70 eli hälytyksen palaaminen lepotilaan. Sitä varten ACProton kiinteissä otsikkokentissä on *alarm status* -tavu niin, että lepo välitetään alkuperäisellä koodilla, mutta Alarm Status -kentässä on tilaksi merkittynä hälytyksen sijasta lepo. Tällä tavalla ATS voi varmistua siitä, että lepoilmoitus koski oikeantyyppistä hälytystä tarkistamalla, että lepoilmoituksen hälytystyypissä oli sama luku kuin aiemmin saapuneessa hälytyksessä. Jos havaittaisiin ristiriita, olisi syytä epäillä että tunkeutuja pyrkii syöttämään järjestelmään väärennetyn lepoilmoituksen. Täman tyyppisen ristiriidan havaitessaan ATS voisi lähettää sabotaasihälytyksen valvomoon.

Alarm status -tavun käyttämiselle on sekin syy, että hälytysten keskitinlaitteen, jonka kanssa ATM0602 liikennöi, on tuettava sellaisiakin protokol-

lia, jotka käyttävät erilaista merkintätapaa. Jossakin toisessa protokollassa lepo ei välttämättä ole laji 70. Alarm status -tavu ilmaisee tällöinkin ATS:lle, onko kyse hälytyksestä vai levosta. Samaten kun myöhemmin lisätään uudenlaisia viestityyppejä hälytyksiä varten kuten tekstiä sisältävä hälytys, on ATS taaksepäin yhteensopiva. Vaikka vanha ATS-versio ei ymmärtäisikään hälytyksen tietosisältöä, se osaa aina päätellä sen hälytykseksi tai levoksi tämän tavun perusteella ja välittää sen eteenpäin, vaikkakin ilman tekstiä.

4.4.1 Salaus ACProtossa

Järjestelmässä käytetty, tietoturvan takaava menetelmä ei perustu viestien salaukseen. Viestit kylläkin salataan, mutta se ei olisi välttämätöntä. Järjestelmässä ei liiku mitään arkaluontoista tietoa. Se, että tietty osoite on tietyn vartiointiliikkeen asiakas, on yleensä kaikkien nähtävillä jo ulko-ovessa olevasta vartiointiliikkeen logosta.

Sen sijaan se, mitä järjestelmässä halutaan ehkäistä, on väärennettyjen ohjaussanomien syöttäminen laitteisiin. Joku saattaisi vaikkapa pyrkiä muuttamaan sen osoitteen, jonne tietyn hälytyssilmukan hälytys lähetetään, väärennetyn asetusviestin avulla. Kyseistä osoitetta kutsutaan hälytyksen *meno-osoitteeksi*. Sen jälkeen olisi mahdollista murtautua kohteeseen vartiointiliikkeen siitä tietämättä. Tämä estetään tuonnempana kuvatulla menettelyllä. Salaus on olemassa ainoastaan siksi, että ilmoituksensiirtoa koskevat standardit sitä vaativat, mikä on tietenkin luonnollista, koska järjestelmien tietoturva on perinteisesti perustunut siihen.

Salauksesta on toisaalta se lisähyöty, että sen avulla voidaan vaikeuttaa palvelunestohyökkäyksen aikaansaamista. Jos hyökkääjä ei osaa muodostaa oikealla tavalla salattuja viestejä, ne hylätään välittömästi niiden saavuttua ilman, että ne aiheuttavat prosessienvälistä kommunikointia ja vievät laitteen resursseja. Lisäksi jo laitteen ethernet-rajapinnan ajuri hylkää kaiken liikenteen, jonka lähettäjän IP-osoite ei vastaa ATS:n osoitetta. Tällä tavalla laitteen ohjelmisto ei edes havaitse tällaisen liikenteen olemassa oloa.

Jos tunkeutuja kuitenkin tietää ATS:n IP-osoitteen, voi hän muokatun ethernet-ajurin avulla väärentää lähetetyissä sanomissa olevan, lähettäjän IP-osoitteen kertovan kentän. IP-osoitteen väärentämiseen perustuvaa hyökkäystä kutsutaan englanniksi nimellä *spoofing attack*. Tällöin seuraavana suojana on viestien salaus, vaikkakin tässä tapauksessa tunkeutuja saa jo kulutettua laitteesta enemmän resursseja, koska ohjelmisto tutkii viestejä. Jos hyökkäys onnistuu, sen hyödyt jäävät kuitenkin kyseenalaisiksi, koska tuloksena on ATS:n suorittama linjavikahälytys, koska laite ei vastaa. Vartijat saapuvat joka tapauksessa tällöin paikalle.

Salausmenetelmä muistuttaa erästä yleisesti tunnettua ohjelmistojen tietoturvamenetelmää. Jos ohjelmisto vaatii toimiakseen merkkijonoavaimen esimerkiksi laittoman kopioinnin estämiseksi, ei tällaisen avaimen tarkistusta kannata tehdä yksinkertaisesti vertaamalla sitä oikeaan avaimeen jossain kohdassa ohjelmakoodia. Jos tunkeutuja onnistuu syöttämään ohjelmalle omia konekielisiä käskyjään esimerkiksi jollakin aiemmin mainitulla menetelmällä, hän voi sijoittaa hyppykäskyn ohittamaan avaimen tarkistuksen. Sen sijaan ohjelmassa olevaa, toiminnan kannalta kriittistä dataa kannattaa säilyttää ohjelmassa salattuna niin, että se saadaan purettua käyttäjän antamalla avaimella. Tällöin mikään yksinkertainen menetelmä kuten hyppykäsky ei hyödytä mitään, koska data pysyy salattuna eikä ohjelma toimi. [21, kpl. 12.2.3].

ACProto käyttää samaa ideaa, ja yksi salausavaimen komponenteista on ATM0602:n ja ATS:n välinen, yhteinen aikakanta. Järjestelmässä halutaan ehkäistä yleisesti käytettyä tunkeutumismenetelmää, jossa salattuja sanomia vakoillaan vaikkapa tunkeutujan hallitsemalla tietokoneella yrityksen lähiverkossa, tallennetaan muistiin ja lähetetään myöhemmin tunkeutumistarkoituksessa. Tällainen viesti tulee kuitenkin hylätyksi siksi, että viestin aikaleima on vanhentunut.

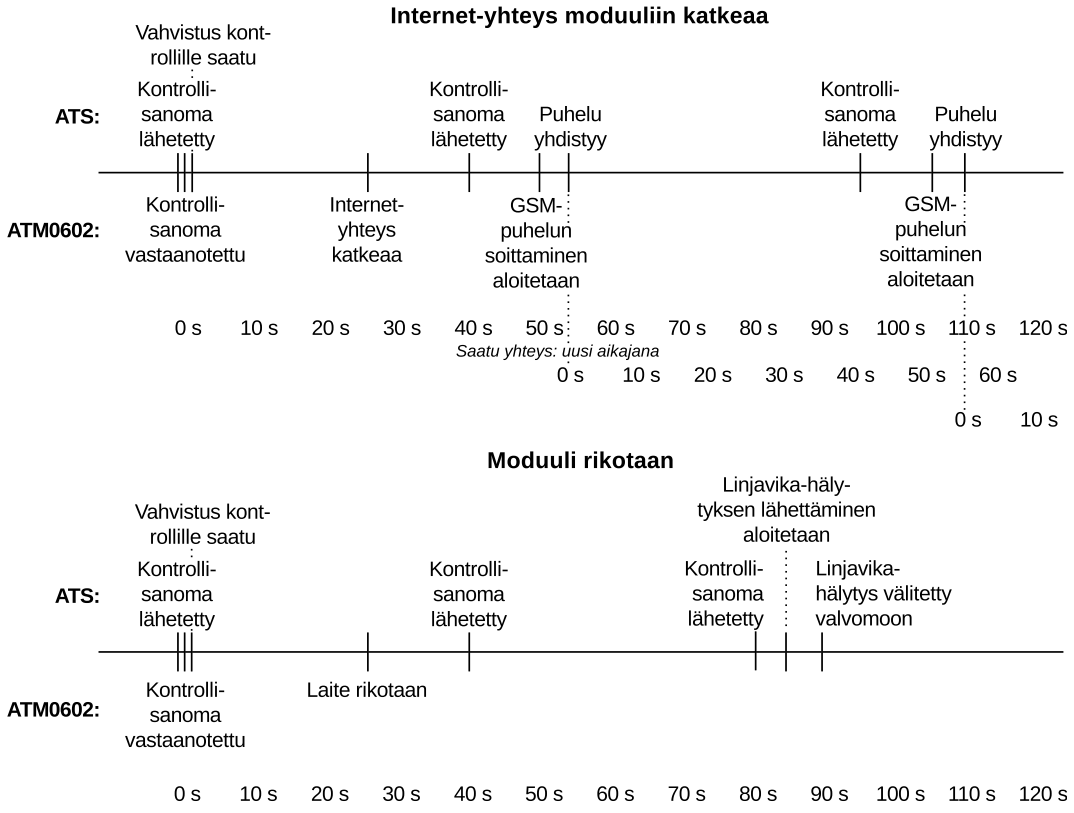
Jos viestin aikaleima olisi sijoitettu ainoastaan tiettyyn kohtaan viestiä, olisi mahdollista, että tunkeutuja saisi sen vaihdettua salauksen sisältäkin, koska kyse on vain muutaman tavun arvaamisesta oikein. Mutta kun aikaleima onkin osa salausalgoritmin avainta, havaittaisiin välittömästi poikkeama viestissä ilmoitetun ja salaamiseen tosiasiassa käytetyn aikaleiman välillä. Viestin salaus ei avautuisi ja viesti hylättäisiin.

4.4.2 Standardien asettamat vaatimukset

Viranomaishyväksytyt hälytyksensiirtolaitteet jakautuvat useaan luokkaan sen mukaan, missä ajassa hälytys siirtyy keskimäärin sekä pahimmassa tapauksessa. Samaten linjavikahälytyksen suurin mahdollinen saapumisaika laitteen rikkoutuessa tai pää- ja varayhteyden katketessa on osa luokitusta. Laitteiden standardinmukaisuutta valvoo Suomessa Inspecta Oy. ATM0602 tarkastettiin yrityksen tiloissa siten, että tarkastaja suoritti pöytäkirjassaan olevat testit niin monta kertaa, että standardin määrittelemien todennäköisyysvaatimusten voitiin osoittaa toteutuneen, kun epäonnistuneita testitapauksia oli alle suurin sallittu määrä.

Ilmoituksensiirtoa käsittelevässä standardissa ilmoituksensiirtolaite jaetaan eri luokkiin kolmen muuttujan perusteella. Näitä ilmaistaan kirjaimilla D, M ja T. Kirjainta seuraa numero, jonka pienin mahdollinen arvo on yksi,

ja suurempi numero tarkoittaa pienempää viivettä ilmoituksen siirtymisessä. Luokka D4 tarkoittaa sitä, että tiedonsiirtoaikojen aritmeettinen keskiarvo olisi korkeintaan 10 sekuntia ja 95 % kaikista tiedonsiirroista tapahtuisi alle 15:ssä sekunnissa. M4:llä puolestaan ilmaistaan, että siirtoajan maksimiarvo on 20 sekuntia. Luokka T5 merkitsee, että linjavikahälytys on valvomossa viimeistään 90 sekunnin kuluttua katkoksen tapahtumisesta ja T6-luokan vastaava aikavaatimus on 20 sekuntia. [18, s. 20] Lähtökohtana työtä aloitettaessa oli, että ATM0602 tulisi olemaan luokissa D4, M4 ja T5. Jos GSM-varayhteyttä ei tarvita, voidaan moduuli asettaa toteuttamaan luokan T6 vaatimukset. GSM-signalointi ei ole riittävän nopeaa T6-luokkaa varten, mutta jos asiakkaalla on erityisen luotettava internet-yhteys, voi tällainen käyttö tulla kyseeseen.



Kuva 4.6: *Aikajana varayhteydelle siirtymisestä internet-yhteyden katketessa sekä linjavikahälytyksen tapahtumisesta, kun murtautuja rikkoo moduulin*

Kuva 4.6 esittää, miten järjestelmä tulisi toimimaan kahdessa eri tilanteessa, kun laitteella on GSM-varayhteys ja se noudattaa siten luokkaa T5 linjavikahälytyksen suhteen. Ylemmässä aikajanassa pääyhteys katkeaa ja laite siirtyy varayhteydelle, eikä linjavikaa siten tapahdu. Jossain vaiheessa pääyhteys sitten myöhemmin palaisi, ja ATS saisi vahvistuksen lähettä-

määnsä kontrollisanomaan. Tällöin palattaisiin normaaliin toimintaan, eli lähettämään kontrollisanomia 40 sekunnin välein.

Alemmassa aikajanassa taas murtautuja saa tavalla tai toisella rikottua moduulin ennen kuin se ehtii lähettää hälytystä. Tällöin seuraa 90 sekunnin kuluessa linjavikahälytys. Kun moduuli myöhemmin vaihdettaisiin toimivaan, se rekisteröityisi ATS:lle pääyhteyden kautta, ja jälleen ATS lähettäisi 40 sekunnin välein kontrollisanomaa.

4.4.3 TCP/IP vaihtoehtoisena yhteyskäytäntönä

Vaikkakin ACProto oli suunniteltu toimimaan optimaalisesti nimenomaan UDP:lla, osattin silti odottaa, että erityisesti yritysverkoissa palomuurit voisivat estää sen käyttämisen. Ongelmana ei niinkään olisi se, että se olisi erikseen estetty, vaan se, että palomuurin refleksiivisyysaika olisi liian lyhyt. Tällä tarkoitetaan seuraavaa: Oletetaan, että yritysverkossa oleva laite lähettää verkon ulkopuoliseen osoitteeseen UDP-datagrammin. Sen vastaanottaja saattaa vastata lähettäjälle toisella datagrammilla. Palomuuri päästää tämän datagrammin yritysverkon sisään vain, jos se saapuu tietyn ajan sisällä siitä osoitteesta, johon yritysverkon sisältä oli aiemmin lähetetty datagrammi. Tätä aikaa kutsutaan refleksiivisyysajaksi. Sen kuluttua umpeen ei datagrammia enää päästetä yritysverkkoon ennen kuin sieltä lähetetään toinen datagrammi. Tämä on tietoturvan vuoksi tärkeää, mutta ACProton kannalta ongelmallista.

Refleksiivisyysajan on oltava vähintään puolet valitusta kontrolliajasta, että kontrolliviesti vielä ohjautuisi palomuurin läpi ATM0602:lle. Ongelma olisi toki voitu kontrolliviestien tapauksessa ratkaista kääntämällä osat toisinpäin niin, että ATM0602 on kontrolliviestin lähettäjä ja ATS taas vahvistuksen. Mutta tällöin ei havaittaisi liian lyhyestä refleksiivisyysajasta seuraavia ongelmia ennen kuin ATS yrittää lähettää ATM0602:lle jonkin komennon, kuten releohjauksen. Tällöin refleksiivisyysaika olisi voinut jo kulua umpeen.

Refleksiivisyysajan ylittymisen seurauksena olisi ohjauksen viivästyminen siihen asti, kunnes ATM0602 jälleen lähettäisi kontrolliviestin ja refleksiivisyysaika alkaisi alusta. Kun ATS on kontrolliviestien lähettäjä, ongelma havaitaan välittömästi, koska ATM0602 siirtyy heti varatielle, koska se ei saanut ensimmäistä kontrolliviestiä. Tällöin tiedetään vaihtaa yhteyskäytännöksi TCP. Siinä nimittäin erikseen ilmoitetaan, milloin liikennöinti alkaa. Tänä aikana taas molempien osapuolien verkkorajapinnan ohjelmakirjastot lähettävät automaattisesti toisilleen yhteyttä ylläpitävää liikennettä, jos yh-

teys on muuten käyttämättömänä. Näin palomuuri on aina selvillä, milloin saapuva data tulee ohjata yritysverkon laitteeseen.

TCP/IP:n käyttämisellä on myös haittansa. Liikennettä tapahtuu kuukaudessa enemmän kuin UDP:llä, koska TCP/IP on sitä monimutkaisempi yhteyskäytäntö. Jos käytössä on langaton pääyhteys, se tarkoittaa suurempaa puhelinlaskua asiakkaalle, jos käytössä on liikenteen määrästä riippuva hinnoittelu. Kukin TCP/IP:lle asetettu laite kuluttaa myös enemmän muistiresursseja ATS:n ohjelmistossa. Näiden asioiden suhteen ei ole tehtävissä mitään.

Aliluvussa 2.2.3 mainittu, SYN-tulvalla toteutettava palvelunestohyökkäys on kuitenkin estettävissä. ATS:n ohjelmallinen palomuuri asetetaan olemaan oletuksena päästämättä SYN-viestejä läpi. Jos se kuitenkin saisi ensin UDP-datagrammin, jossa on oikealla tavalla muotoiltu ja salattu, ACProton määrittelemä TCP-yhteyden avausviesti, se avaisi palomuuriin hetkeksi aukon tästä IP-osoitteesta saapuvalle SYN-viestille. Se ei tällöinkään vielä vastaisi lähettäjälle mitään. Jos sitten lähettäjältä saapuu välittömästi SYN-viesti, vastataan siihen tavanomaisesti ja avataan TCP-yhteys. Mihinkään muuhun ACProton ulkopuoliseen liikenteeseen, kuten ICMP-viesteihin, ATS ei vastaisi. Sen toimintaan ei siis voisi millään tavalla vaikuttaa internetin yli tuntematta tarkasti ACProton määrittelyä. Tämä helpottaisi syyllisen löytämistä palvelunestohyökkäyksen tapahtuessa.

4.4.4 GPRS langattomana pääyhteytenä

Alusta saakka oli päätetty, että ATM0602 tulisi myöhemmin toimimaan myös täysin langattomasti käyttäen pääyhteytenä GPRS-yhteyttä internetiin. Varayhteytenä toimisi GSM-signalointi. Teleoperaattorit eivät anna GPRS:lle takeita yhteysviiveiden tai siirtonopeuden suhteen. Ne riippuvat GSM-verkon tilasta ja ongelmat voidaan välttää vain erillisillä, resursseja hallitsevilla protokollilla, mikä ei ole laitevalmistajan hallinnassa. [10, s. 60] Laitteesta haluttiin tehdä riippumaton käytettävästä teleoperaattorista, joten oli tyydyttävä siihen, että joskus laitteen sijainti estäisi langattoman käytön. On muun muassa havaittu, että lähistöllä oleva koulu on langattomuuden estävä tekijä, välituntisin aiheutuvan GSM-verkon ylikuormittumisen vuoksi.

Myöhemmin kehitetty, 3G-verkossa toimiva laite ratkaisi nämä ongelmat, mutta se perustui eri laitearkkitehtuuriin, eikä siten kuulu työn aihepiiriin. Kyseessä oli sulautettu Linux-alusta ja siinä käytettiin uudelleen ATM0602:n ohjelmistoa viestinvälityksen osalta, mutta muut osat kuten GSM-moduulin laiteajuri, eivät olleet alustan erilaisuuden vuoksi käytettävissä uudelleen.

5. TYÖN ETENEMINEN

Ohjelmointi aloitettiin vuoden 2006 huhtikuussa. Koko projektin ajan käytettiin *CVS*-versionhallintaohjelmaa, ja ensimmäinen versio CVS-projektista oli ainoastaan yksi kehitysympäristön esimerkkiohjelmista, nimettynä uudelleen *AlarmTransmitter.c*-tiedostoksi. Kyseinen ohjelma oli yksinkertainen HTML-käyttöliittymä, jolla voitiin muuttaa kahden ledin tilaa olettaen, että ledit oli kytketty moduulin F-rinnakkaisportin kahteen johtimeen. Rinnakkaisportti on käytännöllinen tapa ohjata maksimissaan kahdeksaa komponenttia porttia kohden, kun komponenteilla on vain kaksi mahdollista tilaa. Rinnakkaisporttiin kirjoitetaan tällöin kahdeksanbittinen luku, jonka kukin bitti ohjaa yhtä komponenttia. Esimerkkitiedosto sisälsi kaiken tarvittavan ohjelmakoodin HTML-käyttöliittymän tekemiseen sekä rinnakkaisporttien käyttämiseen.

5.1 Rekisteröintimenettelyn ja kontrolliviestien toteuttaminen

Kun esimerkkiohjelma saatiin toimimaan, oli seuraava tavoite muokata sen käyttöliittymän sisältöä niin, että sen avulla voitaisiin syöttää laitteen oma IP-osoite, aliverkon peite, yhdyskäytävä sekä ATS:n IP-osoite. Tällöin olisi mahdollista aloittaa protokollan toteuttaminen. Kahden päivän kuluttua tavoite oli saavutettu, ja ATM0602 sekä ATS pystyivät lähettämään UDP-viestejä toisilleen. Seuraavaksi pyrittäisiin siihen, että samassa tilassa ollut, Computec Oy:n valmistama valvomolaite näyttäisi hälytyksen ja sen tiedot, kun ATM0602:n yksi hälytyssilmukka kytkettäisiin oikosulkuun, eli A-rinnakkaisportin tietty johdin yhdistettäisiin sähköiseen maatasoon.

Merkittävä välitavoite saavutettiin vuoden 2006 toukokuun alussa, kun ATM0602 saatiin välittämään onnistuneesti *REG*-sanoma ATS:lle ja lukemaan siltä tullut vastaus, joka on samaa sanomatyyppiä. Kyseessä on ensimmäinen rekisteröintimenettelyn kahdesta askeleesta. Kun laitteet ensimmäisen kerran viestivät keskenään, eivät niiden kellot todennäköisesti käy samaa aikaa. Aliluvussa 4.4.1 mainitusta syystä yhteinen aikakanta on merkittävä osa järjestelmän tietoturvaa. Siksi ATM0602:n kello on ensin saatettava samaan aikaan kuin ATS:n, ja tämä tapahtuu REG-sanomalla. Koska tässä sanomatyypissä ei suoriteta aikakannan tarkistusta, siinä kulkee vain

minimimäärä tietoa, jotta ATS saisi ilmoitettua oman kellonaikansa. Kun ATM0602 saa vastaussanoman, se asettaa oman kellonsa saatuun aikaan. Tästä eteenpäin järjestelmässä käytetään kaikkia tietoturvaominaisuuksia, mukaan lukien aikakannan tarkistusta ja sen mukaan tapahtuvaa salausta. Seuraavaksi ATM0602 siirtyy rekisteröintimenettelyn toiseen vaiheeseen, jossa välitetään muut tarvittavat tiedot. Ne sisältävää sanomaa kutsutaan *REG DATA* -sanomaksi.

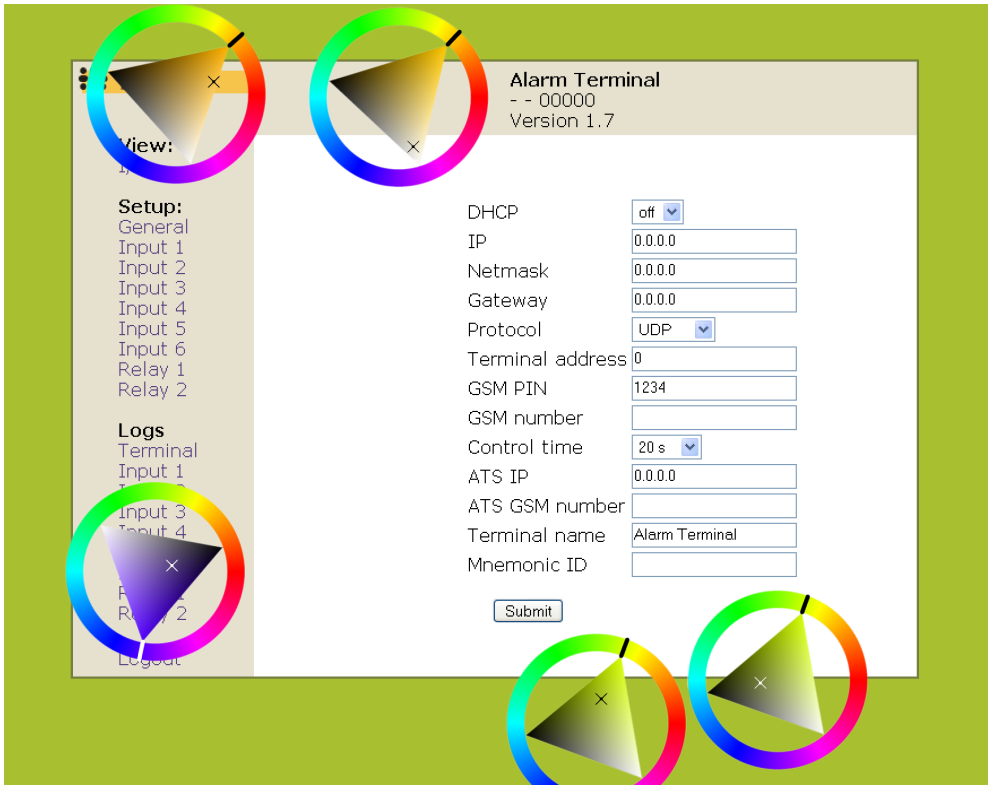
Molempia laitteita kehitettiin yhtäaikaa, ja tässä vaiheessa katsottiin yksinkertaisemmaksi toteuttaa kontrolliviesti eli *CON*. ATS lähettää tätä viestiä säännöllisesti ATM0602:lle ja odottaa saavansa vastauksena *ACK*-sanoman, joka sisältää vahvistetun viestin viestilaskurin numeron. Jos vastausta ei saada ATM0602:n asetuksissa annettuun aikaan mennessä, ATS lähettää linjavikahälytyksen. Samaten ATM0602 aloittaisi uudelleen rekisteröintimenettelyn, jos se ei ole saanut CON-viestiä tänä aikana. CON-viestin toteuttamiseen vaadittiin vain ACK-menettelyn toteuttaminen, kun taas REG DATA -viestiin olisi vaadittu sen lisäksi myös viestien viivästetty suoritus, josta kerrotaan myöhemmin. Toimiva ACK-viestien lähetys sekä uudelleenrekisteröinti, kun havaitaan linjavika, saatiin täysin toimivaksi kahden viikon kuluessa aloittamisesta.

5.2 Graafisen käyttöliittymän ulkoasu

Tässä vaiheessa ohjelmalogiikan kehittäminen jäi tauolle, ja työn alle otettiin laitteen HTML-käyttöliittymä. Toiveena oli, että ATS:ään ehdittäisiin sillä välin toteuttamaan tarvittavat asiat viestien viivästettyyn suorittamiseen, mikä on välttämätöntä, jotta REG DATA -viesti sekä hälytysviesti *ALARM* saataisiin toimimaan.

Käyttöliittymällä on suuri merkitys markkinoinnin kannalta, joten sen ulkonäköön haluttiin kiinnittää huomiota. Vastaavia ilmoituksensiirtolaitteita käytetään tyypillisesti niin kutsutulla terminaaliyhteydellä käyttäen esimerkiksi vanhempien Windows-käyttöjärjestelmäversioiden mukana tulevaa HyperTerminal-ohjelmaa. Graafinen, suoravaikutteinen käyttöliittymä, jossa käsittelyn kohteet ovat näkyvissä ja niitä käytetään osoittimella, on helpommin opittava. Tällainen käyttöliittymä voi näyttää vain tekstiä ja on hitaampi oppia kuin graafinen käyttöliittymä. [14, s. 26] Nykyaikainen käyttöliittymä vähentäisi siten asentajien koulutukseen liittyviä kustannuksia.

Käyttöliittymän kirjasinlajiksi valittiin *Verdana*. Se on erityisesti suunniteltu tietokoneiden näyttöjä, ei tulostusta tai painoa varten, joten se soveltui erinomaisesti tähän käyttötarkoitukseen. Tietokoneiden näytöt on huomioitu Verdanassa esimerkiksi siten, että se on pienessä koossa *sans-serif*-



Kuva 5.1: *Käyttöliittymän värivalinnat*

tyyppinen, kun taas suuremmassa koossa *semi-serif*. [4, s. 30] Näyttöjen erottelutarkkuus on nimittäin tulosteisiin verrattuna hyvin pieni, ja sans-serif-tyyppinen kirjasin on selkeämpi, jos sen piirtämiseen on käytettävissä vain pieni määrä kuvapisteitä. Päätteet kuitenkin parantavat luettavuutta, joten niitä otetaan Verdanassa rajoitetusti käyttöön, kun kirjasimien koko kasvaa.

Värisuunnittelun lähtökohtana oli komplementaarinen värimaailma. Se tarkoittaa että käyttöliittymän päävärien väriarvot valittaisiin väriympyrän vastakkaisista pisteistä. Hashimoton mukaan [9, s. 207] tämä tekee grafiikkaan elävän ja aktiivisen tunnelman. Hän myös mainitsee, että jos väreissä käytetään täyttä värikylläisyyttä, värimaailma on erityisen jännittävä. Tämä ei ole graafisessa käyttöliittymässä tarkoituksenmukaista, joten kaikki siinä esiintyvät värit ovat murrettuja.

Yrityksen logo tuotti oman ongelmansa, koska se ei sovi valittuun värimaailmaan. Onnellisen sattuman seurauksena kuitenkin logon taustavärin eri tavalla murrettu versio on myös Windowsin painikkeissa käytetty väri. Tähän värivalintaan ei olisi voinut vaikuttaa, ja se olisi joka tapauksessa rikkonut värimaailmaa. Käyttämällä tietoisesti samaa sävyä muualla käyttöliittymässä, saadaan se näyttämään yhtenäisemmältä. Näin koko käyttöliittymä rakentuu kolmesta perusväristä. Kuvassa 5.1 näkyy, miten kukin vä-

reistä sijoittuu väriympyrällä ja miten se on murrettu. Väriympyrän keskipisteen kohta sen alla olevassa käyttöliittymäkuvassa merkitsee, mitä väriä se kuvaa.

Lähtökohtana käytettiin vasemmalla olevan valikon painikkeissa esiintyvää violettia sekä taustavärinä toimivaa vihreää. Yrityksen logossa esiintynyt, kellertävä oranssi valittiin eri tavalla murrettuna valikkojen taustaväriksi. Näkyvissä on myös *submit*-nappi, jonka graafisen ulkoasun määrää käyttöjärjestelmä, ja jonka alareunan varjostuksen väri on erittäin lähellä valikkojen taustaväriä.

5.3 Graafisen käyttöliittymän toiminta

Projektin alkuvaiheessa ajateltiin, että käyttöliittymää varten käytettäisiin yhtä kehitysympäristölle tarjolla olleista maksullisista ohjelmakirjastosta, nimeltä *RabbitWeb*. Siitä on myöhemmin tullut maksuton, mutta peruste sen hylkäämiselle ei ollut hinta. Kun ulkoasun määrittelevä HTML-tiedosto oli valmis, kirjaston rajoitteet tulivat ilmeisiksi. Sen avulla oli mahdollista ainoastaan sijoittaa tietoja haluttuihin paikkoihin HTML-tiedostossa, kuten muuttujien arvoja, mutta ei esimerkiksi sijoittaa yhtä HTML-tiedostoa toisen sisään niin, että molemmissa tiedostoissa on ohjelmakoodista peräisin olevaa, dynaamista sisältöä.

RabbitWebiä käytettäessä jokaisen käyttöliittymän sivun olisi pitänyt sisältää koko ulkoasun määrittelevä HTML-sisältö. Sen määrittelevä tiedosto on niin suuri, että tämä lähestymistapa olisi kuluttanut liikaa muistia. Mahdollinen ratkaisu olisi ollut käyttää HTML-kehyksiä, joilla voidaan erottaa sivusta loogisia osia omiin tiedostoihinsa, kuten navigaatiopalkki ja muuttuva osa sisällöstä. Ongelmallista tässä lähestymistavassa on se, että kehyksillä toteutettu sivu ei osaa määrätä omaa kokoaan sen mukaan, kuinka paljon sisältöä kulloinkin sivulla on, vaan lisää kuhunkin osaan vierityspalkin, jos sisältöä on liikaa. Sivun vierittäminen etenkin näppäimistöllä on vaikeaa, koska käyttäjä vierittää helposti eri kehystä kuin hän tarkoitti. Vieritettävä kehys on ensin valittava sarkainnäppäimellä tai napsauttamalla sitä hiiren osoittimella.

HTML-taulukoihin perustuva toteutus välttää nämä ongelmat, jos vain on mahdollista tuoda tiettyyn taulukon soluun toisen HTML-tiedoston sisältö. Tätä varten päätettiin toteuttaa oma merkintäkieli laajentamaan HTML:ää samaan tapaan kuin RabbitWeb laajentaa sitä dynaamisella sisällöllä. Toinen tavoite merkintäkielen suhteen oli, että halutusta kohdasta sivua voitaisiin kutsua haluttua ohjelmiston funktiota sivulla määritellyin parametrein ja sitten tulostaa funktion tuottama sisältö siihen kohtaan sivua. Tällaisia

ratkaisuja varten kehitysympäristössä oli toinen, huomattavasti vaikeammin käytettävä rajapinta, *CGI*.

Käyttöliittymästä haluttiin helposti muokattava. Ohjelmisto olisi ollut liian vaikea ylläpitää jos uuden kentän lisääminen myöhemmin jollekin asetussivulle vaatisi CGI rajapinnan ymmärtämistä. Tämä käy selväksi tutkimalla seuraavassa osiossa esitettyjä, erittäin monimutkaisia tilakoneita, joiden toiminta on ymmärrettävä osatakseen muokata niiden kuvaamia funktioita. Siksi ohjelmiston tuntemat asetukset haluttiin erottaa loogisesti ne toteuttavasta ohjelmakoodista. Liitteenä olevassa ohjelmalistauksessa A.1 esitetään alkuosa vakiotaulukosta, joka määrittelee yleisten asetusten sivun parametrilistan.

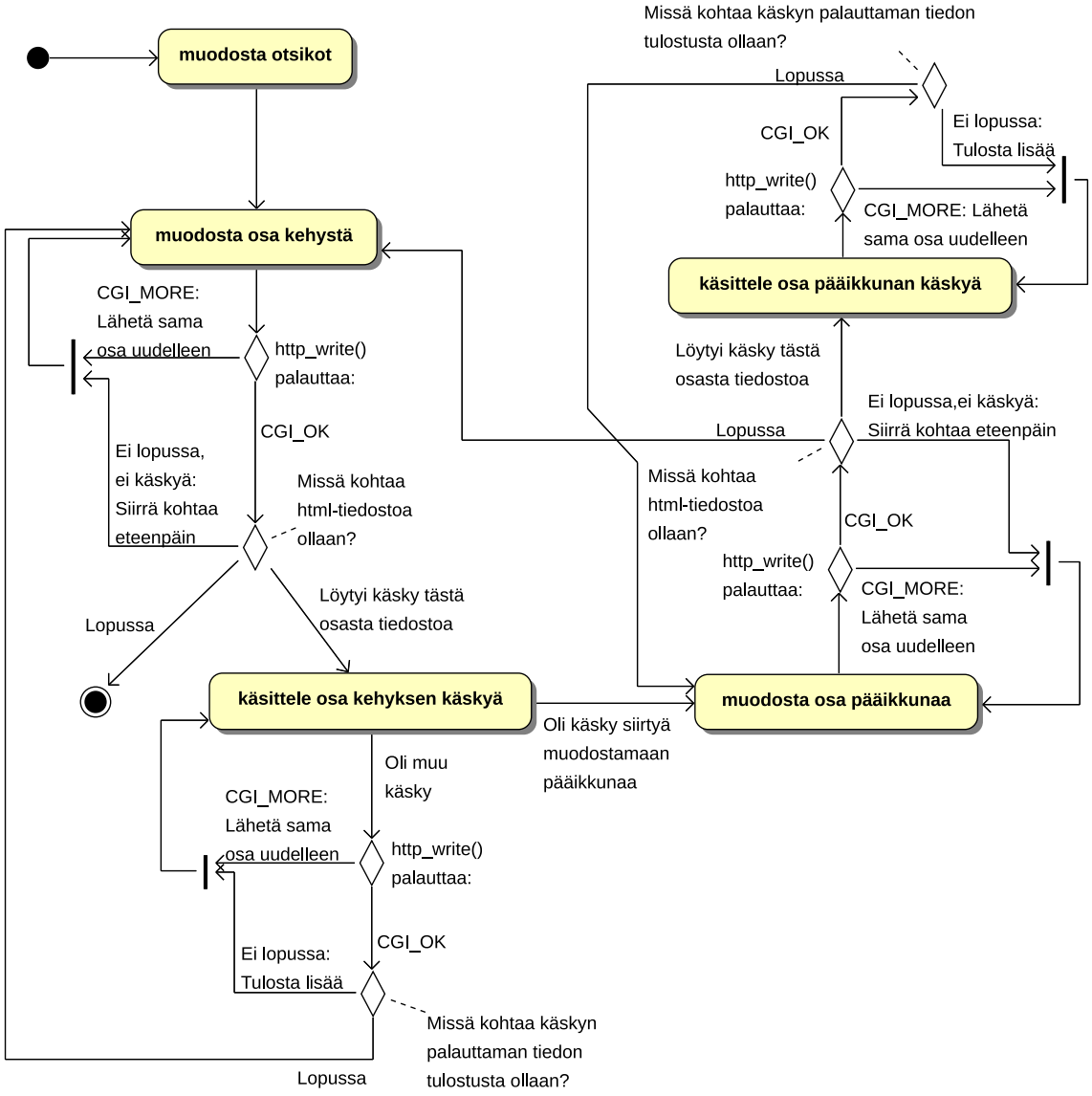
5.3.1 Pageserver-funktio

DynamicC:n CGI-rajapinta toimii niin, että sitä käyttävässä funktiossa käsitellään suoraan käyttäjän selaimelta tulevaa ja sinne lähetettävää tietovirtaa. Halutut moduulin HTTP-palvelimen osoitteet rekisteröidään tiettyyn, CGI-rajapintaa käyttävään ohjelmafunktioon. Kun käyttäjä katsoo selaimellaan tällä tavoin rekisteröityä osoitetta, tehdään ohjelmistossa kutsuja valittuun funktioon. ATM0602:n tapauksessa osoite http://169.254.0.1/pageserver.cgi rekisteröitiin funktiolle nimeltä *pageserver*. Kyseinen IP-osoite kuuluu aina yksityiseen aliverkkoon [16, s. 4] ja se oli siksi turvallista valita laitteen osoitteeksi, kun siihen otetaan paikallinen PPP-yhteys.

Kun pageserver-funktioon tapahtuu kutsu, se saa parametrinaan tietorakenteen, joka sisältää tähän nimenomaiseen tietovirtaan liittyvää tietoa. Tiedot pysyvät samoina funktiokutsusta toiseen niin kauan kuin tietovirtaa riittää, ja kun selain on saanut sivun kokonaan ladattua, tiedot pyyhkiytyvät. Esimerkkejä tietorakenteen tietueista ovat selaimen osoiterivillä annettujen parametrien arvot sekä numeroarvot, jotka merkitsevät tilaa, alitilaa ja alialitilaa. Ohjelman tekijä voi käyttää kolmea viimeksimainittua numeroa niinkuin tarpeelliseksi katsoo. Ainoa kehitysympäristön määräämä asia on, että tietovirran alkaessa ne saavat aina arvon nolla.

CGI-rajapinnalta voi pyytää haluamansa kokoista muistialuetta, jonka tiedot säilyvät tietovirran alusta loppuun, jos on tarpeen tallettaa sellaista tietoa, jota ennalta määritellyt tietueet eivät mahdollista. CGI-rajapinta tukee myös yksinkertaista evästettä, jolla voidaan lähettää käyttäjän selaimelle väliaikaisesti muistissa pidettävä merkkijono, ja jonka sisältö on luettavissa pageserver-funktion parametrin tietueesta. Tarkoitus on muodostaa kullekin käyttäjälle oma merkkijono, jolla käyttäjät saadaan eroteltua toisistaan.

Siten tiedetään, ketkä käyttäjistä ovat kirjautuneet oikealla salasanalla, ja keille on yhä näytettävä kirjautumissivua.



Kuva 5.2: *Tilakone, joka kuvaa pageserver-funktion toimintaa*

Kuvassa 5.2 on esitettynä tapa, jolla käyttöliittymän sivut muodostetaan käyttäen CGI-rajapintaa. Kuten kaaviosta voi havaita, on rajapinnan käyttäminen erittäin monimutkaista. Asioita vaikeuttaa muun muassa se, että tietovirran käsittely ei välttämättä onnistu jokaisella funktiokutsulla, koska kaikkea edellisellä kutsulla muodostettua dataa ei ollut vielä ehditty lähettää käyttäjän selaimelle. Tällöin funktio, jolla data lähetetään, palauttaa arvon *CGI_MORE*. Silloin on seuraavalla funktiokutsulla lähetettävä sama data uudelleen. Jos taas paluuarvoksi saadaan *CGI_OK*, tulee funktion seu-

raavaksi muodostaa uusi osa dataa. Näin edetään, kunnes koko datamäärä on muodostettu yksittäisistä paloista.

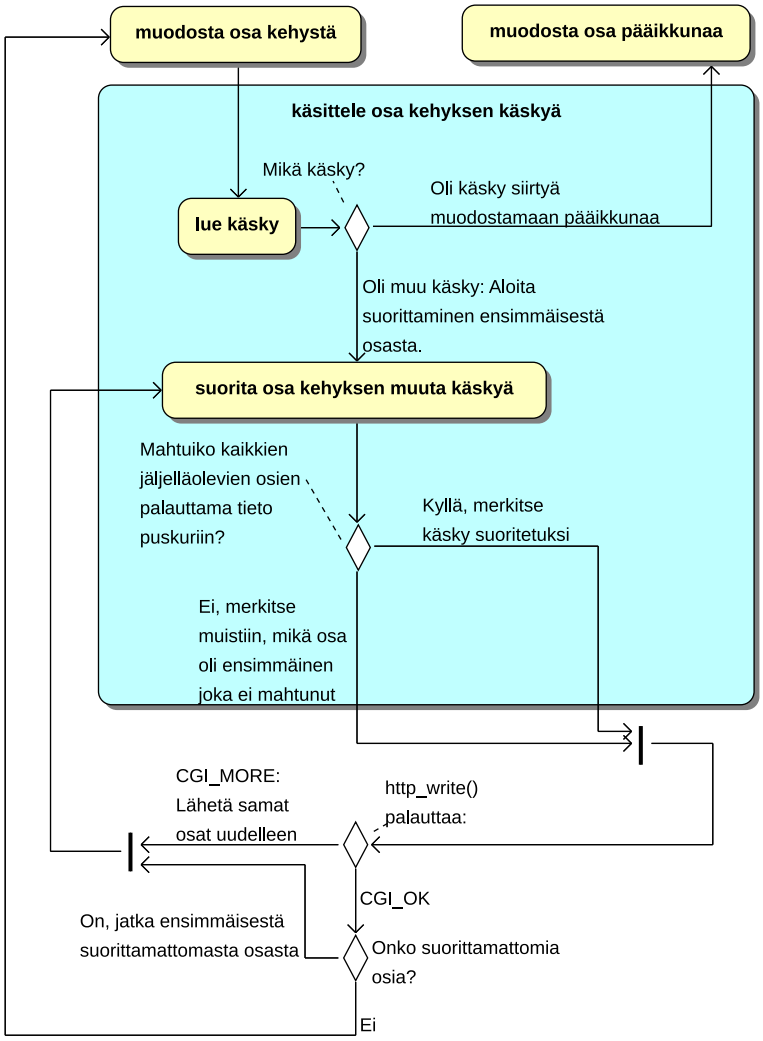
Merkintäkielen komento *$frame$* poikkeaa muista komennoista. Se merkitsee, että tähän kehyksen kohtaan tulee sijoittaa pääikkunan sisältö toisesta tiedostosta. Sen avulla ei kutsuta ohjelmakoodin funktiota, vaan siirretään pageserver-funktio toiseen tilaan, eli käsittelemään pääikkunaa, joka saa myös sisältää komentoja. Toinen *$frame$*-komento ei ole enää siinä tiedostossa sallittu, eli sisäkkäisiä kehyksiä ei tueta.

Yleisellä tasolla pageserver-funktio toimii siten, että se lukee tiedostoa aloittaen lukupisteestä, josta se pitää kirjaa parametrinaan saamassa tietorakenteessa. Se jatkaa lukemista merkki kerrallaan, kopioiden sitä lähetyspuskuriin, kunnes siellä ei ole enää tilaa seuraavalle merkille tai kunnes se kohtaa dollarimerkin, kumpi sitten tapahtuukin ensin. Ensiksimainitussa tapauksessa se ainoastaan siirtää lukupisteen tätä seuraavaan merkkiin. Jälkimmäisessä taas se siirtää lukupisteen dollarimerkkiä seuraavaan merkkiin, ja merkitsee tietorakenteen tilamuuttujaan, että se on nyt lukemassa komentoa. Komento parametreineen ei saa olla pidempi kuin lähetyspuskurin koko, 400 merkkiä.

Siirrettyään lukupistettä funktio lukee seuraavan kutsun yhteydessä tiedoston sisältöä dollarimerkkiin asti, jolloin sillä on puskurissaan komennon nimi ja parametrit, jotka erotetaan välilyönneillä. Se siirtää lukupisteen dollarimerkin jälkeiseen merkkiin, ja suorittaa komennon määräämän funktion. Seuraavalla sivulla olevassa kuvassa 5.3 esitetään käskyjen suorittaminen tarkemmin. Funktio antaa pageserverille merkkijonon, joka ei voi olla lähetyspuskuria pidempi. Jos funktion tarvitsee tuottaa tätä enemmän sisältöä, pidetään pageserverin parametrissa kirjaa siitä, kuinka paljon funktion tuottamasta sisällöstä on jo lähetetty käyttäjän selaimeen.

Jokaisella pageserver-funktion kutsulla lähetetään uusi osa sisältöä, kunnes kaikki on lähetetty. On tosin edelleen mahdollista, että datan lähetysfunktio palauttaa arvon CGI_MORE millä tahansa kutsullaan. Tällöin toimitaan kuten aiemminkin, eli ollaan siirtämättä lukupistettä ja lähetetään sama data seuraavalla kutsulla uudelleen. Kun sitten lähetyspuskurissa on jälleen tilaa, paluuarvoksi saadaan CGI_OK ja lukupistettä siirretään. Hitaasti toimiva internet-yhteys tai käyttäjän selaimen hitaus lisäävät tämän tilanteen todennäköisyyttä.

Tämän jälkeen jatketaan HTML-tiedoston lukemista. Kun ollaan saavuttu pääsivua kuvaavan tiedoston loppuun, palataan siihen kohtaan kehystiedostoa, jossa *$frame$* sijaitsi. Kun taas päästään kehystiedoston loppuun, tietovirta loppuu ja tietorakenteen sisältö kirjoitetaan yli.



Kuva 5.3: *Alitilakone, joka kuvaa tarkemmin pageserver-funktion ”käsittele osa kehyksen käskyä” -tilaa*

5.3.2 Submit-funktio

Pageserver-funktion vastuualue on tietovirta ATM0602-moduulista käyttäjän selaimelle, mutta vastaavasti on käsiteltävä myös vastakkainen suunta, joka muodostuu käyttäjän syötteistä eri sivuille. Tätä varten toteutettiin *submit*-funktio, joka käyttää varsin samankaltaista rajapintaa kuin pageserver. Tietovirtaa käsitellään jälleen osissa, mikä tarkoittaa, että tietyn muuttujan arvoa käsiteltäessä ei vielä tiedetä sitä seuraavien muuttujien arvoja. Tässä suhteessa tapa on siis huomattavasti rajoittuneempi kuin vaikkapa samaan tarkoitukseen suunniteltu PHP-ohjelmointikieli. Toisaalta tämä tapa kuluttaa paljon vähemmän muistia kuin mukaan käännettävä ohjelmointikielen tulkki tarvitsisi.

Liitteenä olevassa kuvassa B.1 on nähtävillä submit-funktion toteutus tilakoneena kuvattuna, pageserverin tapaan. Suoritus muodostuu alkutoimenpiteistä, selaimen lähettämien muuttujien lukemisesta ja niihin liittyvistä toimenpiteistä, sekä jälkitoimenpiteistä. Kun selaimessa painetaan asetussivulla olevaa submit-nappia, selain aloittaa tietovirran lähettämisen. Uuden tietovirran alkaessa tapahtuu kutsu submit-funktioon, ja tällöin suoritetaan alkutoimenpiteet, eli asetetaan tietovirran ajan muistissa säilytettävät muuttujat alkuarvoihinsa, sekä merkitään ajanhetki muistiin. Tämä tehdään siksi, että käyttäjä kirjataan automaattisesti ulos käyttöliittymästä, jos mitään liikennettä moduulin ja selaimen välillä ei ole ollut viiteentoista minuuttiin.

Liikenteenä pidetään myös sivun lataamista, eli pageserver-funktio päivittää myös käyttäjän ulos kirjaamisen ajanhetken. Samalla tarkistetaan, miltä sivulta käyttäjä on lähettämässä tietovirtaa. Jos kyseessä on sisäänkirjautumissivu, niin tarvitsee ainoastaan tarkistaa salasana. Jos se oli oikein, merkitään tälle käyttäjälle lähetetyn evästeen tunnus kirjautuneeksi ja ohjataan käyttäjän selain asetusten etusivulle.

Jos ladattu sivu oli jokin muu kuin sisäänkirjautumissivu, pyydetään selaimelta seuraava osa tietovirtaa. Kukin osa sisältää yhden muuttujan nimen ja arvon. Nämä vastaavat sivulla olevia kenttiä ja alasvetovalikoita. Jollei käyttöliittymää käytetä suoralla sarjakaapeliyhteydellä, kunkin sivun muuttaminen tarvitsee myös kertakäyttöisen muutossalasanan. Jos saatu muuttuja oli muutossalasanakenttään syötetty merkkijono, merkitään se muistiin. Tämä lähestymistapa sisältää sen rajoitteen, että muutossalasanakenttä on aina oltava tietovirran ensimmäinen muuttuja, eli siis ylin kenttä jokaisella sivulla. Rajoitusta voitaisiin kiertää käyttämällä CSS-tyylimäärittelyjä, koska ne sallivat elementtien sijoittamisen riippumatta siitä, missä kohdassa HTML-tiedostoa ne esiintyvät, mutta tähän ei ole ollut tarvetta.

Seuraavan muuttujan saapuessa tarkastetaan, oliko muistiin merkitty muutossalasana oikea. Jos ei, suljetaan käyttöliittymä ja kirjoitetaan istuntoon liittyvät tiedot yli. Käyttäjä ohjataan juuri ennen käyttöliittymän sulkemista sivulle, joka ilmoittaa tapahtuneesta. Tällöin käyttäjän on pyydettävä ATS:ltä uudet muutossalasanat ja käynnistettävä käyttöliittymä uudelleen. Jos salasana taas oli oikea, merkitään salasana tarkistetuksi ja käsitellään saapunut muuttuja.

Kutakin sivua varten on ohjelmakoodiin merkitty, mikä on sen sivun viimeinen muuttuja. Niin kauan kuin ei saada tätä muuttujaa, käsitellään tietovirtaa muuttuja kerrallaan ja tehdään siihen liittyvät toimenpiteet. Useimmissa tapauksissa tämä tarkoittaa asetuksen kirjoittamista muistiin, mutta se saattaa myös tarkoittaa komentoa vaihtaa toinen releistä toiseen asen-

toon. Kun viimeiseksi tiedetty muuttuja on käsitelty, siirrytään jälkitoimenpiteisiin. Lopuksi tarkistetaan vielä, että istunto on kunnossa, eikä edellisestä yhteydestä ole kulunut yli sallittua aikaa.

Käsiteltäessä kutakin muuttujaa on myös merkitty muistiin, että sen arvo on vaihtunut. Näitä muistiinpanoja tarkastellaan jälkitoimenpiteissä. Jotkut muuttujat ovat tietoa, jonka ATM0602 on lähettänyt ATS:lle rekisteröintimenettelyssä. Jos ne muuttuvat, on rekisteröidyttävä uudelleen, jotta ATS saa muuttuneen tiedon. Jos varayhteyttä varten asennetun SIM-kortin PIN-koodi muuttuu, on GSM-moduuli käynnistettävä uudelleen, jotta se saa otettua SIM-kortin käyttöön uudella koodilla.

Lopuksi päivitetään sivu uusiin arvoihin siten, että mahdollisesti virheelliset syötteet ilmaistaan värittämällä kyseinen tekstikenttä punaiseksi. Syöte voi olla virheellinen esimerkiksi siksi, että arvo oli sille muuttujalle sallittujen rajojen ulkopuolella. Punaiseksi värittäminen on toteutettu siten, että ladatessaan sivua uudelleen HTTP-palvelin antaa sille osoiteriviparametrina luvun, jonka kukin arvossa yksi oleva bitti merkitsee, että sen bitin järjestysnumerolla olevassa tekstikentässä on virhe.

Käyttäjän selaimelta saapuvan tietovirran loputtua merkitään annettu muutossalasana käytetyksi. Jos se oli käyttäjän viimeinen muutossalasana, ilmoitetaan käyttäjälle, että hänen on pyydettävä lisää muutossalasanoja jos aikoo vielä tehdä muutoksia, ja suljetaan käyttöliittymä.

Kun ATS lähettää etäkäyttöä varten muutossalasanat ATM0602:lle, niiden tietoturva toteutuu aliluvussa 5.5 tarkemmin kuvatulla menettelyllä. Lyhyesti sanottuna, vaikka tunkeutuja saisi viestien tarkan muotoilun ja salausmenetelmän selville esimerkiksi varastamalla ohjelmiston lähdekoodin Telcont Oy:n palvelimelta, ei olisi silti mahdollista lähettää väärennettyjä salasanaviestejä ATM0602:lle ja vaihtaa niiden avulla asetuksia.

5.4 Tapahtumapuskuri

Käyttöliittymä oli siinä määrin toimiva vuoden 2006 kesäkuun alussa, että projektissa keskityttiin jälleen ensimmäisen välitavoitteen saavuttamiseen, eli hälytysviestien lähettämiseen. Käyttöliittymän ohjelmakoodista tosin oli korjailtava siinä ilmeneviä virheitä vielä pitkään tämän jälkeen. Erityisesti virhetilanteet, jotka seurasivat http_write-funktion CGI_MORE paluuarvosta, olivat vaikeita korjata. Koska tämä tilanne seuraa HTTP-palvelimen lähetyspuskurin täyttymisestä, olivat silloin suoritettaviin toimenpiteisiin liittyvät virheet vaikeita toistaa.

Tähän asti oli riittänyt, että laite tietää seuraavan tapahtuman ja sen ajanhetken. Ohjelmalogiikka asetettiin kullakin logiikkatehtävän suorituskerral-

la odottamaan tapahtumaa viestijonosta tietyllä aikarajalla. Odottaminen voi johtaa kahteen eri toimenpidesarjaan: Jos viesti saapuu ennen aikarajaa, reagoidaan sen mukaisesti. Jos taas aikaraja ylittyy, tehdään kyseisen viestin saapumisaikarajan ylittymiseen liittyvät toimenpiteet. Esimerkkinä mainittakoon REG-viestin lähettäminen. Jos vastausviesti saapuu ajoissa, merkitään laite rekisteröityneeksi. Jos taas aikaraja ylittyy, suoritetaan REG-viestin saapumisaikarajan ylittymiseen liittyvä toimenpide, joka on REG-viestin uudelleenlähetys.

Seuraavaksi laitteen haluttiin voivan pitää kirjaa suuremmasta määrästä tapahtumia. Ajatellaanpa tilannetta, jossa laite lähettäisi hälytyksen, ja ennen vahvistusviestin saamista laitteelle saapuisikin kontrolliviesti. Toisaalta olisi osattava sijoittaa seuraavan kontrolliviestin odotus oikeaan ajanhetkeen, ja toisaalta edelleen muistettava odottaa vahvistusviestiä hälytykseen siltä varalta, että hälytysviestille tarvitsee suorittaa uusintalähetys.

Liitteenä olevassa kuvassa C.1 on nähtävillä, miten ongelma ratkaistiin. Se tosin kuvaa lopullista toteutusta. Aluksi viestejä ei tiivistetty yhdeksi, jatkuvaksi muistialueeksi muistin säästämiseksi, vaan kullekin tapahtumalle oli oma, kiinteän mittainen lokero. Vasta kun tapahtumiin liittyvien datamäärien erot kasvoivat suuriksi, ja siten lokeron olisi tarvinut olla kohtuuttoman kokoinen, jouduttiin toteuttamaan kuvassa esitetty ratkaisu. Muutos toteutettiin siten, että funktiot, joilla puskuria käsiteltiin, pysyivät koko ajan samanlaisina funktioiden kutsujan kannalta.

Tapahtumat sijoitetaan aikajärjestykseen ja jokaiseen liitetään aika edellisestä tapahtumasta kyseiseen tapahtumaan. Ensimmäisen tapahtuman aika on suhteessa nykyhetkeen. Tapahtumilla on kiinteän mittainen otsikko-osa sekä vaihteleva määrä dataa, jonka pituus nähdään tietystä otsikkokentästä. Näin tiedetään koko tapahtuman datan pituus ja siten se, mistä seuraava alkio alkaa. Tutkimalla tapahtumia järjestyksessä alusta loppua kohden, löydetään etsittävä tapahtuma. Tapahtumaa voidaan etsiä esimerkiksi viestinumeron, joka on yksi kiinteistä otsikkokentistä, perusteella. Puskurissa pidetään myös kirjaa siitä, kuinka monta tavua siitä on käytössä, ja näin tunnistetaan se, milloin ollaan löydetty puskurin viimeinen tapahtuma.

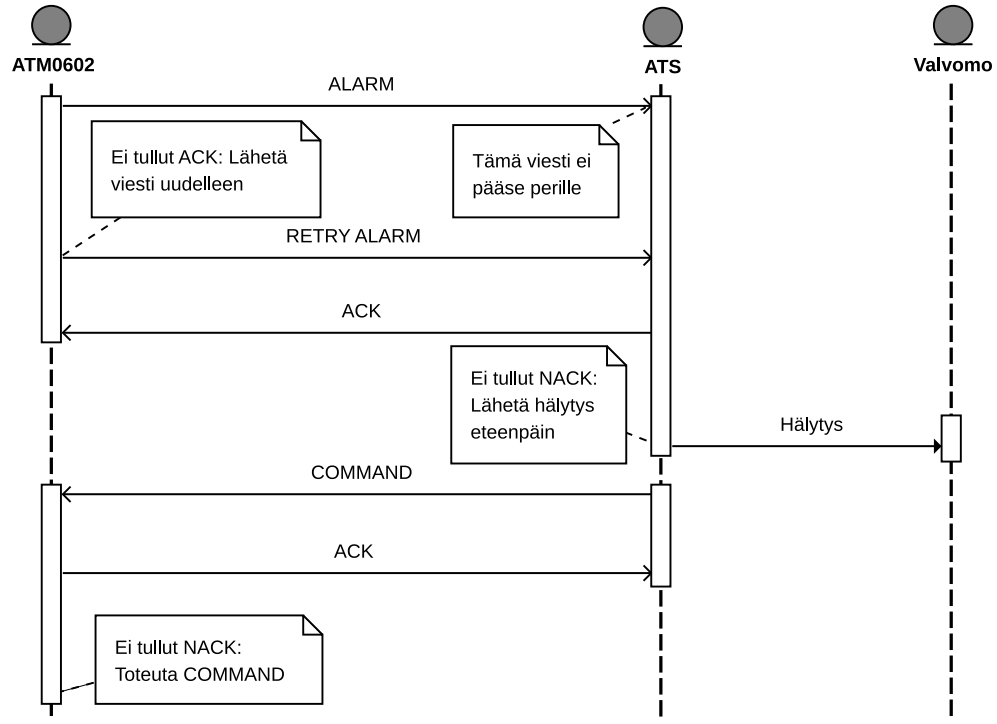
Kun puskuriin sijoitetaan tapahtuma, rajapintafunktio saa parametrinaan ajan nykyhetkestä uuteen tapahtumaan. Funktio etsii sille oikean paikan niin, että kaikki tapahtumat pysyvät aikajärjestyksessä, ja päivittää seuraavan tapahtuman ajan, koska se merkitsee nyt aikaeroa äsken sijoitettuun tapahtumaan. Logiikkatehtävän suorituskerroilla oli myös päivitettävä tapahtumien ajat, koska nykyhetki oli siirtynyt odotettaessa eteenpäin. Tässä ilmenee etu, joka saavutetaan kun ajat ovat suhteessa edelliseen tapahtu-

maan, eivätkä nykyhetkeen. Aikaa päivitettäessä ei tarvitse koskaan muuttaa useamman kuin yhden, eli ensimmäisen, tapahtuman aikaa. Tapahtumien aikavälit toisiinsa nähden kun eivät tällöin muutu. Aluksi tätä optimointimenetelmää ei oivallettu, vaan kaikki ajat olivat suhteessa nykyhetkeen, jolloin jokainen tapahtuma oli päivitettävä uuteen aikaan.

5.5 Viestiliikenteen lopullinen toteutus

Ensimmäiseksi vuorossa oli hälytysviestien toteuttaminen. Tässä vaiheessa hälytyssilmukoiden tarkkailu tapahtui vielä varsin yksinkertaisesti. Tila tarkistettiin sadan millisekunnin välein, ja jos se poikkesi edellisessä tarkistuksessa saadusta arvosta, katsottiin tilamuutos tapahtuneeksi. Kytkinvärähtelyiden suodattaminen jäi myöhemmäksi.

Kuvassa 5.4 olevan tapahtumasekvenssikaavion ensimmäinen vaihe selventää hälytysviestien välitystä. Toistosanomien lähetysväli langallisella internet-yhteydellä liikennöitäessä on yksi sekunti. Silmukan tilamuutoksen tapahduttua asetetaan tapahtumapuskuriin toistosanoman lähetystapahtuma sekunnin päähän nykyhetkestä.



Kuva 5.4: *Tapahtumasekvenssikaavio hälytyksen lähettämisestä ATM0602:lta ATS:lle, jonka jälkeen ATS lähettää komennon ATM0602:lle*

Jokaisen logiikkatehtävän suorituskerran päätteeksi katsotaan tapahtumapuskurin ensimmäisen tapahtuman aika, joka on siis suhteessa nykyhetkeen. Logiikkatehtävä asetetaan odottamaan, että jotain saapuisi sen viestijonoon. Odotuksen aikarajaksi asetetaan äsken tarkistettu aika. Kun tapahtumatehtävä seuraavan kerran pääsee suoritukseen, viestijonon odotusfunktion paluuarvosta nähdään, alkoiko suoritus siksi, että saatiin viesti, vai siksi, että aikaraja ylittyi. Jälkimmäisessä tapauksessa katsotaan jälleen tapahtumajonon ensimmäinen tapahtuma.

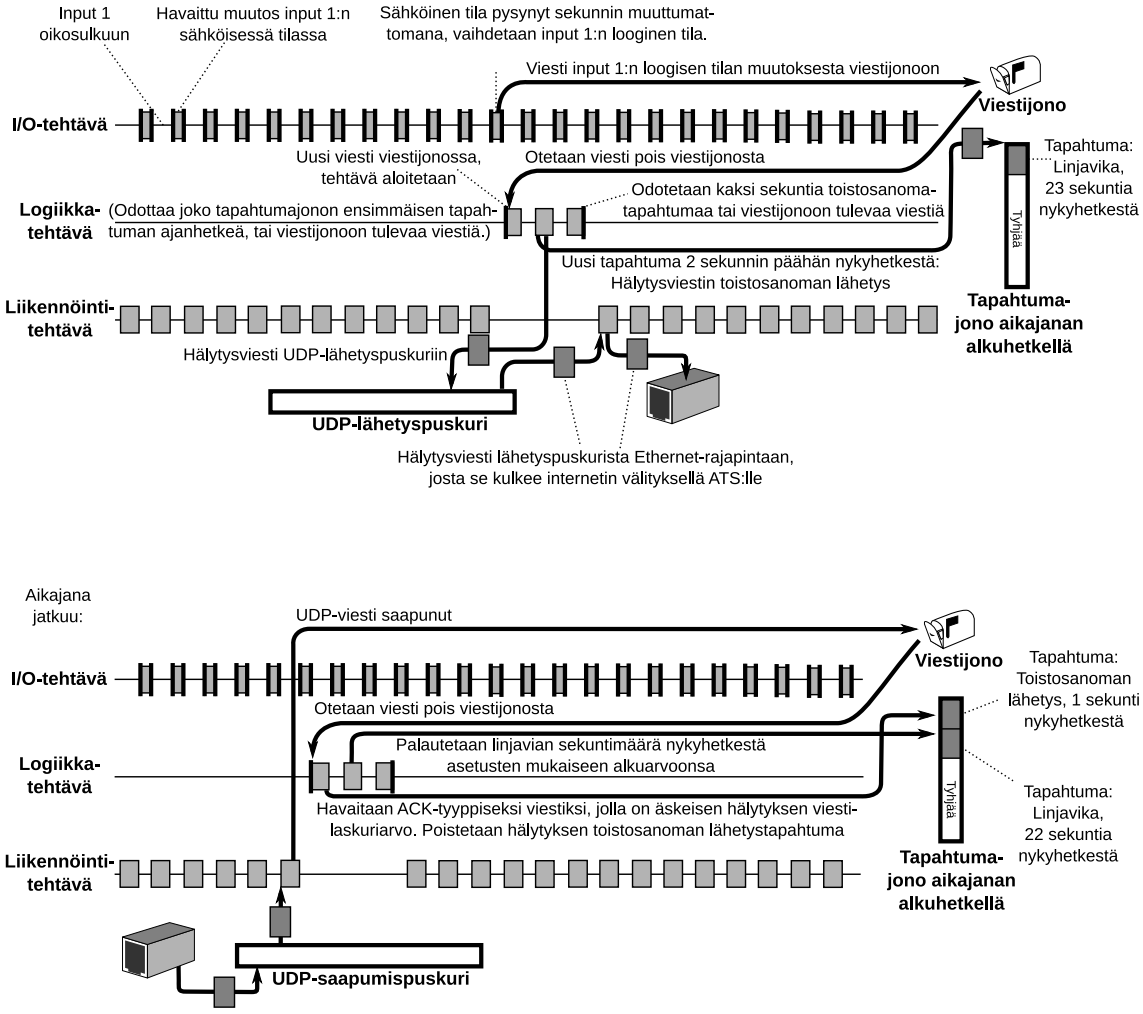
5.5.1 Hälytysviestien alustava toiminta

Hälytysviestien lähetysmenettelyn tapauksessa ensimmäisen alkion havaitaan olevan viestin uusintalähetystapahtuma. Alkion vaihtuvanmittaisessa datassa on kaikki hälytysviestin muodostamiseen tarvittava tieto. Tämä data annetaan viestin lähetysfunktiolle, joka muodostaa viestin ja lähettää sen ATS:lle. Seuraavaksi uusi lähetystapahtuma sijoitetaan jälleen sekunnin päähän nykyhetkestä. Tällöin se ei välttämättä ole enää ensimmäinen tapahtuma jonossa, vaan ensimmäiseksi saattaa päätyä vaikkapa jonkin toisen viestin uudelleenlähetys.

Uudelleenlähetyksiä ei tehdä loputtomasti. Tapahtuman otsikkotiedoissa on luku, joka kuvaa sitä, kuinka monta kertaa se on toistettu. Kahdeksannen uudelleenlähetyksien kohdalla oletetaan, että pääyhteys ATS:ään on katkennut ja ryhdytään toimenpiteisiin. Jos varayhteys on saatavilla, siirrytään käyttämään sitä, aloittaen soittamalla välittömästi ATS:lle. Jos ei, aloitetaan rekisteröintimenettely. Näin toimitaan siksi, että syynä viestiliikenteen katkeamiseen saattaa olla se, että ATM0602:n ja ATS:n kellot ovat päätyneet siinä määrin eri aikaan, että ne hylkäävät toistensa viestit tietoturvamenetelmän vuoksi. Rekisteröinti palauttaa kellot samaan aikaan.

Kun ATS saa hälytysviestin, se lähettää sille vahvistuksen. Liikennöintitehtävä ottaa vahvistuksen vastaan verkkorajapinnasta ja purkaa salauksen, jos viestin aikaleima on riittävän lähellä laitteen kellonaikaa. Samalla se varmistaa tarkistussummista, että viesti on virheetön. Sitten se sijoittaa saamansa viestin logiikkatehtävän viestijonoon.

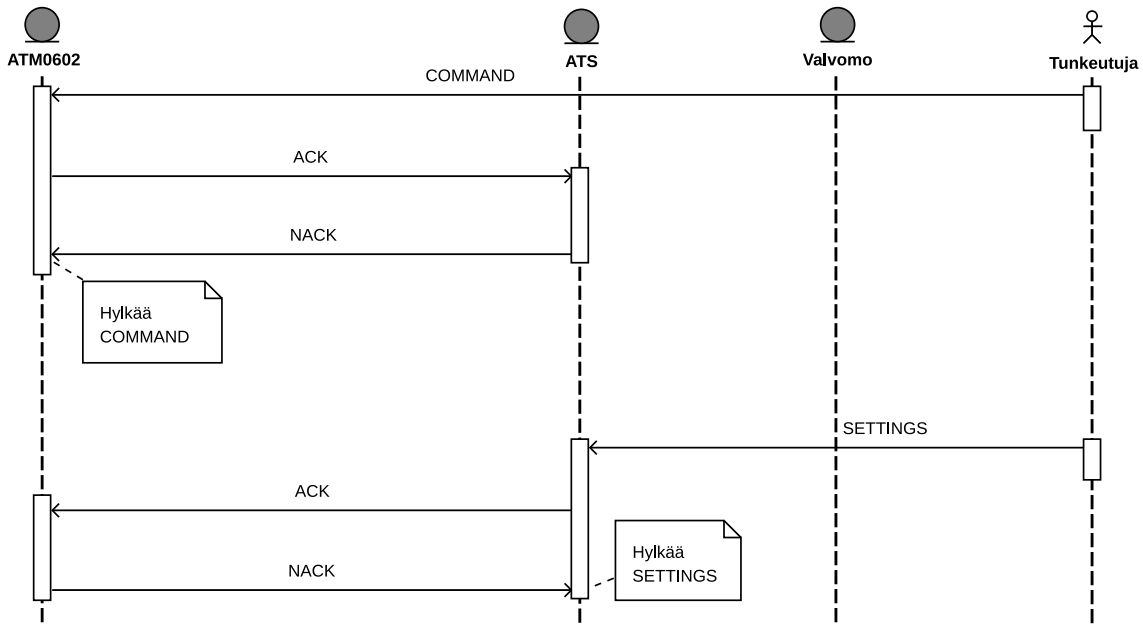
Saadessaan vahvistusviestin logiikkatehtävän kutsuma tapahtuman poistofunktio käy tapahtumapuskurin läpi. Se etsii sellaista uudelleenlähetystapahtumaa, jolla on vahvistusviestissä saatu viestilaskurin arvo. Jos tapahtuma löydetään, se poistetaan puskurista liitteenä olevassa kuvassa C.1 esitetyllä tavalla, siirtäen myöhempiä tapahtumia taaksepäin niin, että muistialueelle ei jää tyhjää tilaa.



Kuva 5.5: *Esimerkki tehtävien toiminnasta, kun ATM0602:ssa tapahtuu sähköinen muutos hälytystulossa yksi ja linjavikaan on tapahtumahetkellä 23 sekuntia*

Kuvassa 5.5 esitetään, miten tilamuutos havaitaan ja hälytys lähetetään sekä vahvistus vastaanotetaan kunkin tehtävän osalta. Se esittää lopullista toteutusta, eli kytkinvärähtelyjen suodatus on kuvassa mukana, vaikka sitä ei ollut vielä tässä vaiheessa toteutettu. Vahvistusviesti oletetaan kuvassa saapuvan sekunnin kuluttua tapahtumasta. Kuvan alempi aikaraja alkaa tästä ajanhetkestä.

Kuvaa 5.5 tulee lukea niin, että seurataan nuolien tulo- ja lähtöpisteitä vasemmalta oikelle sikäli, kun ne koskettavat tehtäviä kuvaavia aikajanoja. Tapahtumajonossa näkyvä linjavikatapahtuma on jonossa aina ja sen ajanhetkeä siirretään eteenpäin, kun ATS:ltä saadaan dataa. Tämä kertoo tapahtumien aikajärjestyksen. Hälytysten välittäminen saatiin toimimaan vuoden 2006 kesäkuun puolessavälissä. Näihin aikoihin jouduttiin myös tekemään paljon korjauksia käyttöliittymään. Viestiliikenteen kaikki vaaditut ominai-



Kuva 5.6: *Tapahtumasekvenssikaavio tietoturvan toteutumisesta, kun tunkeutuja pyrkii vaikuttamaan ATM0602:n ja ATS:n toimintaan*

suudet, kun käytössä on langallinen internet sekä protokollana UDP, oli valmis noin kolmen viikon kuluttua.

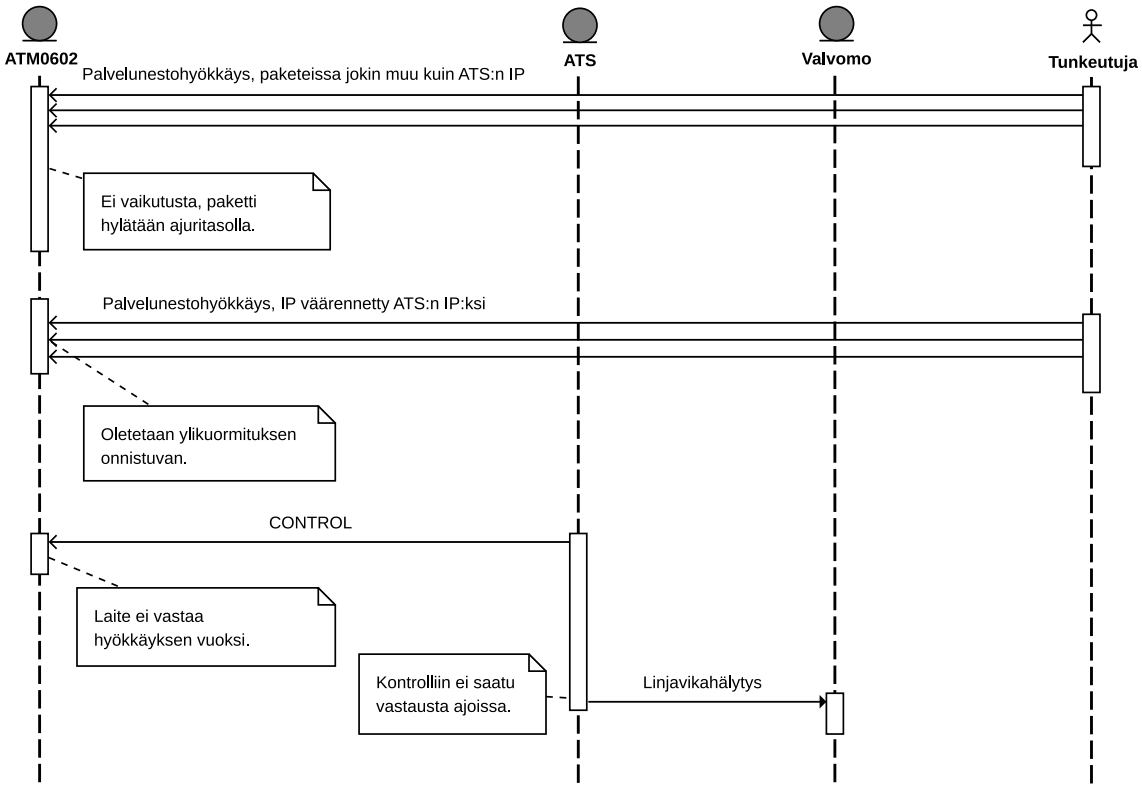
5.5.2 Viestien viivästetty suoritus

Sivulla 46 esitetyn kuvan 5.4 sekvenssikaavion jälkimmäinen tapahtuma kuvaa viestin viivästettyä suoritusta, joka on tärkeä tietoturvaominaisuus järjestelmässä. Kuvasta 5.6 ilmenee syy tähän. Kun ATM0602 tai ATS saa viestin, se ei ryhdy toimenpiteisiin välittömästi. Se lähettää ainoastaan vahvistusviestin, jonka jälkeen se jää odottamaan yhteystyypistä riippuvaksi ajaksi. Langallisella internet-yhteydellä aika on yksi sekunti, ja langattomalla kaksi. Jos tänä aikana saapuu *NACK*-viesti, mikä tarkoittaa, että liikenteen toinen osapuoli on saanut vahvistuksen viestiin, jota se ei ole lähettänyt, odottava viesti jätetään suorittamatta.

Koska ATS:ään ja ATM0602:een on asennusvaiheessa syötetty vastapuolen IP-osoite, tunkeutuja ei voi ohjata vastausta muualle väärentämällä viestin lähettäjän osoitetta. Vastauksen kohde katsotaan aina laitteen flash-muistissa olevista tiedoista, ei saapuneesta viestistä. Kuvassa on esitetty kaksi epäonnistunutta yritystä vaikuttaa järjestelmän toimintaan.

Ensimmäisessä tapauksessa yritetään syötettää väärennetty komento ATM0602:lle, kuten releen avaaminen, jos siihen on liitetty sähkölukko. Jälkimmäisessä taas pyritään vaihtamaan ATS:llä muistissa olevia asetuksia

kuten linjavikahälytyksen kohdetta. Jos tämä onnistuisi, voisi tunkeutuja yrittää murtautua kohteeseen ja rikkoa laitteen ennen kuin se ehtii lähettää hälytyksen, jolloin linjavika ei saapuisi vartiointiliikkeeseen virheellisten tietojen vuoksi.



Kuva 5.7: *Tapahtumasekvenssikaavio menetelmästä, jolla ATM0602:ssa on varauduttu palvelunestohyökkäyksiin*

Myös palvelunestohyökkäykset ovat uhka julkisessa IP-verkossa. Kuvassa 5.7 näkyy, miten niihin on ohjelmistossa varauduttu. Jos tunkeutuja ei tiedä ATS:n IP-osoitetta, ei hyökkäyksellä ole mitään vaikutusta. Laitteen Ethernet-rajapinnan ajuria on muokattu niin, että se ei ota vastaan viestejä, jotka eivät saavu sille annetusta IP-osoitteesta.

Tämä sulkee pois suurimman osan hyökkäyksiä, koska ATS:n IP-osoitteen selville saaminen vaatii sisäpiirin tietoa, ja viestien lähettäjän IP-osoitteen väärentäminen vaatii tietotekniikkaosaamista. Jos kuitenkin tämä onnistuu ja oletetaan, että ATM0602 saadaan ylikuormitettua niin, että se lakkaa liikennöimästä, ei tälläkään saavuteta merkittävää hyötyä kohteeseen murtautumisen kannalta. Linjavikahälytys tapahtuu asetuksissa määritetyn ajan kuluttua sekvenssikaaviossa kuvatulla tavalla.

5.6 Toimintojen lisääminen valmiiseen arkkitehtuuriin

Kun arkkitehtuuri oli osoitettu toimivaksi, oli aika lisätä toimintoja. Käyttöliittymään toteutettiin lisää asetuksia, kuten GSM-varayhteyden puhelinnumerot sekä SIM-kortin PIN-koodin asetus. Myös hälytyssilmukkojen ja releohjauksien asetukset lisättiin käyttöliittymään. Eräs mainitsemisen arvoinen näistä oli hälytysviive, koska tätä käytettiin kytkinvärähtelyiden suodattamiseen.

Ohjelmassa erotettiin toisistaan silmukan sähköinen ja looginen tila. Sähköinen tila on se tila, joka on havaittu viimeisimmällä tarkistuskerralla. Aina kun se muuttuu, tallennetaan sen yhteyteen muutoksen ajanhetki. Aina kun jonkin silmukan sähköinen tila havaitaan toiseksi kuin looginen tila, verrataan tallennettua ajanhetkeä nykyhetkeen. Kun aikaero kasvaa vähintään yhtä suureksi kuin asetuksissa oleva hälytysviive, vaihdetaan myös looginen tila. Samalla aiheutuu hälytys, jonka silmukoita tarkkaileva I/O-tehtävä välittää logiikkatehtävän viestijonoon. Viivettä voidaan hyödyntää myös muihin tarkoituksiin. Esimerkiksi liiketunnistusta tekevä valvontakamera voitaisiin asettaa lähettämään hälytys vasta, kun se on havainnut liikettä tietyn aikaa.

Hälytyssilmukoiden asetuksiin toteutettiin myös meno-osoitteet, eli hälytysviestien kohteena olevat osoitteet. Kullakin valvomolla on oma osoitteensa, joka koostuu ylä- ja alaverkosta sekä pisteosoitteesta, sivulla 10 kuvatulla tavalla. Tämä tarkoitti sitä, että hälytysviestien osalta oli erotettava toisistaan toistosanoman ja kopiosanoman käsitteet. Suuri määrä meno-osoitteita pelkästään toistosanomilla toteutettuna johtaisi ongelmiin: jos käytettäisiin kahdeksaa meno-osoitetta ja jokainen kopiosanoma kasvattaisi toistolaskuria, se päätyisi maksimiarvoonsa, vaikka kaikki viestit olisi välitetty onnistuneesti perille.

5.6.1 Hälytysten sarjallistaminen

Hälytysviestejä toteutettaessa ilmeni puute ohjelmalogiikan suunnittelussa. Muiden viestien kuin hälytysten kohdalla viestien saapumisjärjestys ei merkinnyt mitään. Oli siis tehokkainta hoitaa useiden viestien lähetys rinnakkain siten, että ei odotettu vahvistusta yhteen viestiin ennen kuin toista jo lähetettiin. Näin taattiin se, että jokin vähemmän tärkeä viestiliikenne ei viivyttäisi hälytysviestiä. Kaikki tarvittava lähetettiin kerralla, ja toistettiin sitten sanomia sitä mukaa, kuin niiden aikarajat ylittyivät. Vastapuoli saattoi sitten tarvittaessa poimia tärkeimmät sanomat ensin käsittelyyn. Mutta hälytysviestien kohdalla havaittiin riski, jota ei ollut huomioitu. Jos silmukan

tila vaihtuisi ensin hälytykseen ja sitten nopeasti lepoon, saattaisi hälytyksen sisältävä viesti kadota matkalla, mutta levon sisältävä taas päästä ensimmäisellä yrityksellä perille. Sitten hälytys toistettaisiin, ja nyt valvomolla olisikin viimeisimpänä tilanaan virheellisesti hälytys eikä lepo.

Mahdollinen ratkaisu tähän olisi ollut järjestellä viestejä laskurin arvon perusteella, mutta tämä olisi ollut vaivalloista toteuttaa ATS:ssä. Huomattavasti yksinkertaisempaa kokonaisuuden kannalta oli tehdä hälytysviestistä poikkeustapaus siten, että niitä voisi olla kerrallaan lähetyksessä vain yksi. Vasta kun se saisi vahvistuksen, otettaisiin toinen hälytystapahtuma käsittelyyn. Kunhan tapahtumat siirrettäisiin tapahtumapuskuriin siinä järjestyksessä kuin ne havaittiin, olisi helppoa taata, että ATS saa ne oikeassa järjestyksessä.

Oikean saapumisjärjestyksen takaamiseksi tehtiin silmukan tilamuutoksille oma puskurinsa. Kun muutos tapahtuu, tarkistetaan ensin, onko tapahtumapuskurissa jo muutostapahtumaa. Jos ei ole, siirretään tapahtuma suoraan sinne. Jos taas on, laitetaan se viimeiseksi alkioksi tilamuutospuskuriin, joka toteutettiin rengaspuskurina. Kun taas tilamuutokseen saadaan ATS:ltä vahvistusviesti viimeiseen lähetettävään kopiosanomaan, poistetaan ensin sitä vastaava alkio tapahtumapuskurista ja sen jälkeen otetaan ensimmäinen alkio tilamuutospuskurista tapahtumapuskuriin, jos siellä sellainen on. Sitten alkaa tämän tilamuutoksen toisto- ja kopiosanomien lähetys. Näin edetään kunnes yhtään tilamuutosta ei ole kummassakaan puskurissa.

5.6.2 Oikea rekisteröintimenettely ja muistinkäytön optimointi

Rekisteröitymismenettely jaettiin kahdeksi viestiksi niin, että heikosti salatussa REG-viestissä kuljetettiin minimimäärä tietoa, ja loput tiedot sijoitettiin tavanomaisesti salattuun ja viivästettyyn REG DATA -viestiin. Samoin toteutettiin laitteen tehdasasetuksiin palauttaminen, kun painikkeita painetaan sivulla 26 kuvatulla tavalla. Tätä toteutettaessa myös ylitettiin laitteen muistin rajat ensimmäisen kerran. Lisää vapaata muistia saatiin siirtämällä kaikki HTML-sisältö laitteen ylämuistiin. Se on käyttömuistia huomattavasti suurempi muistialue, jota ei kuitenkaan voi käyttää ohjelmassa suoraan, vaan josta on ensin haettava sisältöä käyttömuistiin ohjelmointiympäristön tarjoamilla rajapintafunktioilla.

Myös debug-tekstirivejä lyhennettiin ja poistettiin mahdollisuuksien mukaan. Ohjekirjasta myös selvisi tapa lisätä käytössä olevaa muistia jakamalla ohjelmakoodi ja käyttömuisti omiin muistialueisiinsa. Tämän sivuvaikutus oli, että ohjelmakoodi ei voisi enää refleksiivisesti muuttaa itseään, mutta tä-

tä ominaisuutta ei pidetty tärkeänä. Se myös sulki pois sivulla 17 mainitun kekomuistin manipuloimiseen perustuvan tunkeutumismenetelmän.

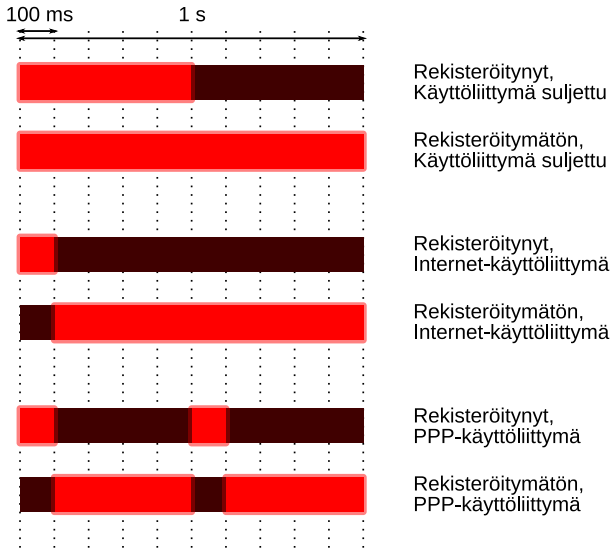
5.6.3 Ohjaus- ja kyselyviestit

Samaan aikaan toteutettiin ne ohjaus- ja kyselyviestit, jotka oli projektin alussa määritelty tuettaviksi. Kaikki edellämainittu ohjelmistokehitys tapahtui limittäin sen mukaan, mikä parhaiten soveltui ATS:n sen hetkiseen tilanteeseen. Kun ATS:lle tehtiin ohjelmakoodin refaktorointia tai korjattiin virheitä, oli käytännöllisintä keskittyä käyttöliittymään ja ATM0602:n ohjelmakoodin refaktorointiin. Kun taas korjaukset saatiin valmiiksi, oli aika jatkaa viestiliikenteen toteuttamista. Ohjaus- ja kyselyviestit vaativat keskittymistä molempiin laitteisiin.

Kullakin ohjaus- ja kyselyviestillä on oma tunnuksensa, joka on luku väliltä 0—255. Lisäksi ne voivat sisältää 0—232 tavua dataa. Tosin ei voida taata, että vastaanottava laite tukee täysimittaista dataa. Varmoja voidaan olla ainoastaan siitä, että se ymmärtää kaksi datatavua. Kyselyviestien avulla on mahdollista selvittää laitteen ohjelmaversio sekä pyytää laitteelta tapahtumaloki, jos kyselyn suorittava laite tukee sen näyttämistä. Laitteen hälytyssilmukoilta taas voi selvittää, ovatko ne hälytyksessä vai levossa tai ylipäänsä toiminnassa. Releiltä voi kysyä ovatko ne auki vai kiinni.

Ohjausviesteillä voidaan asettaa hälytyssilmukka pois käytöstä tai vaihtaa aktiivinen tila, eli se, merkitseekö silmukan auki vai kiinni oleminen hälytystä. Rele voidaan ohjata kiinni ja auki, sekä sille voidaan antaa impulssiohjaus, jossa relettä pidetään kiinni sekunnin ajan ja sitten se avataan. Hälytyssilmukoille ja releille on mahdollista antaa ohjaus- ja kyselyviestejä vain niistä osoitteista, jotka on erikseen mainittu kyseisen silmukan tai releen asetuksissa, joten tällä menetelmällä ei voida tehdä haittaa valvomosta käsin vaikkapa kilpailijan omistamalle laitteelle. Jos lähettäjän osoitetta ei tunnisteta, lähetetään vastauksena ”kielletty toimenpide” -tyyppinen viesti. Tähän pisteeseen päästiin projektissa vuoden 2006 elokuussa. Ohjelmakoodi merkittiin ensimmäiseksi vakaaksi ohjelmaversioksi ja perustettiin kehitysversiolle uusi haara.

Myöhemmin havaittiin esimerkiksi viestiliikenteen joskus pysähtyvän ilman näkyvää syytä. Ongelma saatiin täysin poistettua vasta, kun ennen jokaista viestin lähetystä tehtiin ARP-taulukon tarkistus kehitysympäristön rajapintafunktiolla ja tarvittaessa asetettiin laite tekemään erikseen ARP-pyyntö verkkoon. Sellainenkin puute havaittiin mukana tulevissa ohjelmakirjastoissa, että ensimmäinen tämän jälkeen lähetettävä UDP-viesti ei koskaan pääse perille. Siksi oli ohjelmoitava laite yrittämään vain otsikkoken-



Kuva 5.8: *ATM0602-moduulin tilan ilmoittaminen Status LED:n avulla*

tät sisältävän UDP-viestin lähetystä aina ARP-kyselyn jälkeen, niin että myöhemmin lähetettävä, todellista dataa sisältävä viesti pääsisi perille asti. DHCP oli huomattavasti ongelmallisempi kuin staattisen IP-osoitteen käyttäminen kehitysympäristön erikoisuuksien vuoksi, ja se saatiin toimimaan täysin luotettavasti vasta lokakuussa.

5.6.4 Status-ledin toiminta

Yksi erillinen työvaihe tänä ajanjaksona on erotettavissa muista. Se on status-ledin toiminnan suunnittelu ja toteutus. Tavoitteena oli ilmaista kaksi asiaa yhdellä kertaa sen avulla. Toisaalta se, onko laite omasta näkökulmastaan rekisteröitynyt ATS:lle ja toisaalta se, onko käyttöliittymä aktiivisena. Käyttöliittymän suhteen haluttiin lisäksi esittää, onko se avattu sarjakaapeliyhteydelle vai etäyhteydelle internetin läpi.

Esitettävänä oli siis kaksi toisistaan riippumatonta muuttujaa, josta toisella voi olla kaksi arvoa ja toisella kolme. Olisi siis suunniteltava kuusi eri sekvenssiä, joiden avulla kaikki arvojen yhdistelmät olisi esitettävissä niin, että kysymyksiin ”onko laite rekisteröitynyt?” ja ”mikä on käyttöliittymän tila?” saataisiin helposti vastaukset.

Kuvassa 5.8 näkyy valittu ratkaisu. Kysymys ”onko ledi 50 % tai enemmän ajasta pimeänä?” vastaa siihen, onko laite rekisteröityneenä. Jos on pimeänä, on se rekisteröitynyt, ja toisinpäin. Muissa tapauksissa käyttöliittymän tilan määräämä sekvenssi yksinkertaisesti vaihdetaan käänteiseksi, mutta kuvan ylimmän sekvenssin käänteinen versio olisi mahdotonta erottaa alku-

peräisestä. Siten rekisteröitymätön tila, jossa käyttöliittymä on suljettuna, esitetään pitämällä lediä jatkuvasti päällä. Tämä sekvenssi valittiin tähän tilaan siksi, että todellisessa käyttötilanteessa sen havaitseminen merkitsisi aina ongelmaa, johon olisi puututtava. Jos kuitenkin ohjelmisto olisi jäänyt määrittelemättömään tilaan vastoin kaikkia pyrkimyksiä estää tätä, se voisi näkyä samankaltaisena ledin jatkuvana palamisena, kun I/O-tehtävä ei pääse vaihtamaan sen tilaa. Vaikka ohjelmiston vahtikoira-ajastin onkin luotettava tapa estää tämä ongelma ohjelmavirheiden aiheuttamana, on silti mahdollista että virransyötössä tapahtuva äkillinen jännitteen väliaikainen laskeminen jättää suorittimen tilaan, josta sitä ei saada palaamaan millään ohjelmistoteknisellä keinolla.

Ledin säännöllinen vilkkuminen kertoo siis nyt kaksi asiaa: Toisaalta sen, että ohjelmisto on toiminnassa ja toisaalta sen, että hälytyksensiirto on kunnossa. Siten, jos laitteesta vastaava asentaja soittaa jonkin ongelman tai kysymyksen vuoksi, yksinkertaisella kysymyksellä ”vilkkuuko status-led?” saadaan jo paljon tärkeää tietoa ilman, että eri sekvenssejä tarvitsee tarkoin kuvailla. Status-ledin toimintaa alettiin suunnitella vuoden 2006 elokuun alussa ja toimiva ohjelmakoodi oli valmis seuraavana päivänä.

5.6.5 Settings-viestit

Asetusviestejä alettiin toteuttaa vuoden 2006 lokakuun lopussa. ATM0602:n haluttiin tallettavan kaikkien asetusten nykyiset arvot sekä muutoshistorian ATS:lle. Arvot välitettäisiin ensimmäisen rekisteröitymisen yhteydessä tehdasasetuksiin palautuksen ja sitä seuraavan asetusten syöttämisen jälkeen. Sitten ATM0602 lähettäisi muuttuneita asetuksia aina sitä mukaa, kuin niitä käyttöliittymästä vaihdellaan. Laitteen tulisi myös muistaa ne muutokset, jotka on tehty sen ollessa rekisteröitymättömässä tilassa, ja lähettää ne laitteen rekisteröitymisen jälkeen.

Ajatuksena oli, että jos ATM0602 vikaantuisi ja olisi vaihdettava toiseen laitteeseen, sen kaikki asetukset olisi palautettavissa ATS:ltä takaisin. Ainoastaan internet-yhteyden asetukset olisi syötettävä käsin, jotta laitteet ylipäänsä voisivat viestiä keskenään. Kuhunkin asetuksen muutokseen liitettäisiin ATS:llä aikaleima niin, että tietyllä ajanhetkellä olleet asetukset olisi mahdollista palauttaa laitteeseen ATS:ltä käsin. Ominaisuus oli valmis marraskuussa. Settings-viestit muodostetaan varsin yksinkertaisella, ahneella algoritmilla.

Kussakin yksittäisessä viestissä on tilaa 232 merkille dataa. Algoritmi käy lähettämättömien asetusten listaa läpi alusta loppuun, ja kokeilee, mahtuisiko viestiin vielä yksi asetus ja sen arvo. Niin kauan kuin mahtuu, muodos-

tetaan viestiä. Kun seuraava asetus ei enää mahdu, keskeytetään listassa eteneminen ja sijoitetaan viestin lähetys tapahtumapuskuriin. Sitten aloitetaan jälleen tyhjästä settings-viestistä ja jatketaan viestien muodostamista, kunnes kaikki parametrit on merkitty lähetetyiksi.

Kun viesti sitten saa vahvistuksen, käydään tapahtumapuskuriin tallennetun viestin sisällön perusteella läpi, mitkä asetukset se sisälsi. Nämä asetukset merkitään lähetetyiksi. Lähetettyjen asetusten listaa päädyttiin lopulta säilyttämään flash-muistissa, vaikka niiden muuttaminen kuluttaakin flash-muistin rajallisia uudelleenkirjoituskertoja. Aluksi listaa pidettiin laitteen paristovarmennetussa RAM-muistissa, mutta sitten todettiin, että pariston loppuminen aiheuttaisi liian vakavaa haittaa laitteen toiminnalle.

5.6.6 Virtuaalisarjaportti, Modbus-liitäntä ja salaus

Telcont Oy:n toiset työntekijät olivat keskittyneet vuoden 2006 marraskuusta eteenpäin kahteen ominaisuuteen, joista kumpikin osoittautui lopulta tarpeettomaksi. Kirjoitushetkellä ei ole ainuttakaan asiakasta, joka olisi koskaan käyttänyt niitä. Toinen oli virtuaalisarjaportti, jonka avulla voitiin liittää sarjaportin avulla liikennöivä laite ATM0602:een niin, että Telcont Oy:n valmistaman laiteajurin avulla se näkyisi PC-tietokoneelle kuin laite olisi liitetty suoraan siihen. Vastaavasti tietokoneeseen liitetty laite käyttäytyisi kuin se olisi liitetty ATM0602:n sarjaporttiin moduulin ohjelmiston kannalta. Kun ohjelmistossa luettaisiin tiettyä suorittimen sarjaporttia tai kirjoitettaisiin siihen, tapahtuisi liikennöinti tosiasiassa PC:hen liitetyn laitteen kanssa. Ajatuksena oli, että näin voitaisiin ohjata vaikkapa kääntyvän valvontakameran moottoria, tai kirjoittaa tapahtumalokia ATM0602-moduulista tietokoneen kirjoittimelle.

Toinen ominaisuus oli Modbus-liitäntä. Sen avulla laitteen hälytyssilmukoiden määrä saatiin yli sataan ja releitäkin oli liitettynä kymmeniä erillisen Modbus master-laitteen kautta, joka puolestaan oli liitetty ATM0602:n sarjaporttiin. Ratkaisu olisi ollut asiakkaalle kallis, koska se olisi vaatinut master-laitteen ostamista kolmannelta osapuolelta.

Nykyään ATM0602:n rinnalla Telcont Oy:n tuotevalikoimassa oleva, tehokkaampi versio asiakaslaitteesta nimeltään *MTM* (Multi-Transmitter Module) käyttää huomattavasti halvempaa ratkaisua. Sen käyttämä lisäkortti on Telcont Oy:n suunnittelema ja käyttää sen omaa protokollaa. ATM0602:een ei ole enää tarjolla laajennusmahdollisuutta, koska Modbus-liitännän ylläpito ohjelmaversioissa lopetettiin kannattamattomana. Kirjoittajan osuus näiden kahden ominaisuuden suhteen oli integroida ne ohjelmalogiikkaan sekä käyttöliittymään.

Salausta alettiin toteuttaa joulukuussa. Sen yksityiskohtia ei voida tässä turvallisuussyistä kuvailla. Kyseessä on itse kehitetty salausmenetelmä jossa yhteinen aikakanta on osa salausavainta, minkä syy on kerrottu aliluvussa 4.4.1. Lähetettäessä viestiä se viedään salausfunktion läpi, ja viestin saapuessa taas purkufunktion. Ohjelmalogiikan toimintaan salauksen olemassaolo ei vaikuta. Salauksen toteuttamiseen kului yksi viikko.

5.7 GSM-varayhteys

GSM-signalointiin ei ollut tarjolla valmista ohjelmakirjastoa, joten Telcont Oy ryhtyi kehittämään sellaista vuoden 2007 helmikuussa. Kirjoittaja ei osallistunut kirjaston kehitystyöhön, vaan ainoastaan sen integrointiin ohjelmalogiikan kanssa. Ensimmäinen käyttökelpoinen versio kirjastosta valmistui maaliskuussa. Tätä ennen kontrolliviestin saapumisen takarajan määrittelevä alkio tapahtumapuskurissa oli yksinkertaisesti sijoitettu nykyhetkestä *kontrolliajan* verran eteenpäin vähennettynä vakiolla.

Kontrolliajalla tarkoitetaan asetuksissa määriteltyä aikaa, jossa tähän laitteeseen liittyvän linjavian on viimeistään oltava välitettynä valvomoon siitä hetkestä, kun yhteys katkeaa. Hieman ennen tätä hetkeä laite havaitsi, että edellisen viestin saapumisesta ATS:ltä on kestänyt liian pitkään ja aloitti rekiteröitymismenettelyn. Sitten kun yhteyskatko oli ohi, rekisteröityminen onnistui ja oltiin jälleen valmiita siirtämään hälytyksiä.

Nyt kuitenkin tämä menettely oli jaettava kahteen erilliseen ajanjaksoon. Jos kuvitteellisen aikajanan nollapiste on edellisen viestin saapumishetki, lähettää ATS seuraavan kontrolliviestin hetkellä *kontrolliväli* / 2 – 5 sekuntia, ellei muuta liikennettä tapahdu. Puolet kontrolliajasta on siis se hetki, kun lähetyksestä on kulunut viisi sekuntia, ja kontrolliviestin olisi siten pitänyt jo saapua perille. Tapahtumapuskurin alkio asetettiinkin nyt siis kontrollijakson puoleenväliin. Sillä hetkellä voidaan jo todeta, että jotain on vialla, ja toisaalta on vielä aikaa ryhtyä toimenpiteisiin toisin kuin kontrollijakson lopussa, jolloin linjavikahälytys on jo valvomossa.

Jos GSM-varayhteys ei ole käytössä, asetetaan yksinkertaisesti uusi tapahtuma saman ajanjakson päähän ja kasvatetaan tapahtuman toistolaskuria yhdellä. Kun sitten sama tapahtuma havaitaan nollaa suuremmalla toistolaskurin arvolla, tiedetään, että kontrollijakso on päättynyt ja ATS on lähettänyt linjavian eikä siis enää lähetä kontrolliviestejä ennen seuraavaa rekisteröitymistä. Näin varayhteydetön toiminta pysyi ulkoisesti samanlaisena kuin ennenkin.

Jos varayhteys on olemassa, suoritetaan jakson puolivälissä soitto GSM-moduulilla ATS:n puhelinnumeroon. Tätä kutsutaan *kontrollisoitoksi*. Saa-

dessaan soiton siitä numerosta, jonka ATS tietää kuuluvan ATM0602:lle sen rekisteröitymisen yhteydessä saamien asetusten perusteella, ATS siirtää linjavikahälytyksen ajanhetkeä eteenpäin samaan tapaan kuin saadessaan vahvistuksen kontrolliviestiin. ATS ainoastaan välittää valvomoon ilmoituksen siitä, että tietty ATM0602-moduuli on nyt siirtynyt varayhteydelle. Laitteen rekisteröityessä jälleen pääyhteyden kautta lähetetään tästäkin ilmoitus. Jos soitto jää jossain vaiheessa tapahtumatta ennen linjavikahälytyksen ajanhetkeä, kyseessä on linjavika, jonka ATS välittää hälytyksenä valvomoon. Myös rekisteröintimenettelyn aikana suoritetaan soittoja samalla aikavälillä, ellei olla jossain vaiheessa saatu ATS:ltä kieltoa soittaa uudelleen.

GSM-signaloinnin avulla saadaan kaikki tarvittava tieto molemmille osapuolille ilman, että sitä tarvitsee välittää puhelussa. Soiton vastaanottaja voi joko vastaanottaa tai hylätä puhelun, ja soittaja saa tiedon siitä, kumpi asioista tapahtui. Ajurissa tämä tapahtuu niin, että sen annettua *AT-komentona* soittokäskyn GSM-moduulille sarjaportin kautta, moduuli antaa standardien määrittelemän vastauksen, josta puhelun tilanne saadaan selville. AT-komennot ovat modeemina toimivien laitteiden yhteinen käskykanta, jota kukin valmistaja voi myös laajentaa omilla, vain kyseisessä laitteessa toimivilla komennoillaan.

Jos ATS hylkää puhelun, se on merkki ATM0602:lle, että varayhteys on kunnossa. Jos se taas ottaa puhelun vastaan ja sitten sulkee sen välittömästi, se tarkoittaa, että ATS on jo lähettänyt linjavian. Tällöin ATM0602 lakkaa soittamasta, koska siitä ei olisi enää hyötyä. Se yrittää silti välittää hälytyksen GSM-datapuheluna tämän jälkeenkin, jos sellainen tapahtuu. Yhteyden toiminnassaolon ilmaisemisesta puhelun hylkäämisen avulla on se etu, että siitä ei koidu laskentaa asiakkaan puhelinlaskuun. Ainoastaan hälytysten siirtämisestä ja soittamisen lopettamisesta puhelun vastaamisen avulla koituu kustannuksia.

Aina, kun ATM0602:ssa tapahtuu hälytyssilmukan tilamuutos silloin, kun se ei ole rekisteröityneenä pääyhteydellä, se soittaa GSM-datapuhelun ja välittää hälytykseen liittyvän datan sillä tavoin. Ohjelmalogiikan kannalta tämä ei poikkea merkittävästi viestin välittämisestä pääyhteyden avulla. Ainoa poikkeus on, että puhelun soittaminen on ensin aloitettava GSM-ajurin rajapintafunktion avulla.

Kun puhelu on yhdistynyt, ajuri sijoittaa tästä kertovan viestin ohjelmalogiikan viestijonoon. Sitten ohjelmalogiikka antaa välitettävän datan sekä vastaanottajan puhelinnumeron parametrina toiselle GSM-ajurin rajapintafunktiolle. Siirrettyään datan tai tiedonvälityksen epäonnistuttua ajuri lähettää jälleen tästä kertovan viestin, ja ohjelmalogiikka joko välittää uuden

tilamuutosviestin tai kutsuu puhelun lopetusfunktiota, mikäli välitettävää ei enää ole.

Varayhteys toimi kaikilta osin vuoden 2007 huhtikuussa. Tällä välin löydettiin ja korjattiin myös muita virheitä, kuten puutteita Telcont Oy:n ennen tätä projektia ohjelmoimasta serial flash -muistin käyttöä helpottamaan tarkoitetusta kirjastosta. Samoin löydettiin pieniä virheitä viestiliikenteestä, kuten ongelma viestilaskurin arvoissa, jos rekisteröitymisen yhteydessä tuli matkalla viipynyt, toinen viesti. Se tuli hylätyksi rekisteröitymättömyyden vuoksi, mutta sitten sen saapuessa uudelleen, kun se olisi oltu valmiina käsittelemään, se tulkittiinkin toistosanomaksi ja hylättiin jälleen.

5.8 TCP/IP-yhteyden toteutus

TCP/IP:n käyttäminen pääyhteyden yhteyskäytäntönä otettiin työn alle vuoden 2007 toukokuussa. Työ jaettiin jälleen kahteen osaan, jossa Telcont Oy:ssa kehitettiin viestien lähetys- ja vastaanottorajapintoja TCP/IP:n ollessa käytössä, ja kirjoittajan tehtävänä oli integroida ne ohjelmalogiikkaan. Tämä päästiin aloittamaan kahden viikon kuluttua.

Aiemmin rekisteröitymismenettely oli hyvin yksinkertainen. Ohjelmalogiikka oli tilakone, jolla oli kolme tilaa: *Rekisteröitymismenettelyn ensimmäinen osa ei lähetetty*, *toinen osa ei lähetetty* sekä *laite rekisteröitynyt*. Jos jossain tilassa tapahtuu virhe, kuten liian monta epäonnistunutta yritystä, palataan ensimmäiseen. Muuten tiloja edetään järjestyksessä eteenpäin aina, kun toimenpide suoritetaan onnistuneesti. UDP:n ollessa yhteyskäytäntönä tiloissa ei tarvitse kuin lähettää viestejä, mikä on muutenkin ohjelmalogiikan asia. Liikennöintitehtävä ainoastaan huolehtii, että *tcp_tick*-funktiota suoritetaan säännöllisesti. Nimi on hieman harhaanjohtava, koska funktio huolehtii myös UDP-liikenteestä. Tämä funktio vastaa lähetysjonoon sijoitettujen viestien lähettämisestä ja TCP/IP:n tapauksessa siihen kuuluvista, automaattisista toistosanomista.

TCP/IP kuitenkin vaati myös yhteyden avaamisen tässä vaiheessa. Sitä ei voinut jättää ohjelmalogiikan tehtäväksi, koska verkkorajapintaan liittyvät funktiot eivät ole vapaakäyntisiä, eli niitä ei voida käyttää yhtäaikaa useammasta tehtävästä. Liikennöintitehtävän ja logiikkatehtävän olisi siis tehtävä yhteistyötä, koska näiden funktioiden käyttö oli liikennöintitehtävän asia. Tämä tapahtui ottamalla uusi MicroC/OS-II-kirjaston palvelu käyttöön: lippujen odottaminen. Samaan tapaan kuin tehtävä voitiin asettaa odottamaan viestijonoon saapuvaa viestiä, se voi myös odottaa, että tietyn muuttujan tietty lippubitti vaihtaa arvoa. Se voidaan myös asettaa odottamaan, että toinen kahdesta bitistä vaihtuu, tai molemmat.

Nyt siis lisättiin tilakoneeseen neljäs tila, joka olisi ennen kaikkia muita, eli *yhteys ei avattu*. UDP:tä käytettäessä ohjelmalogiikka siirtyisi aina suoraan tämän yli, mutta TCP/IP:llä se asettaisi lippubitin, jonka merkitys on *avaa TCP/IP yhteys*. Sitten se jäisi odottamaan, että joko tämän bitin tila vaihtuu takaisin, tai sitten toisen bitin, joka merkitsee *TCP/IP-yhteyden avaaminen epäonnistui*, arvo vaihtuu.

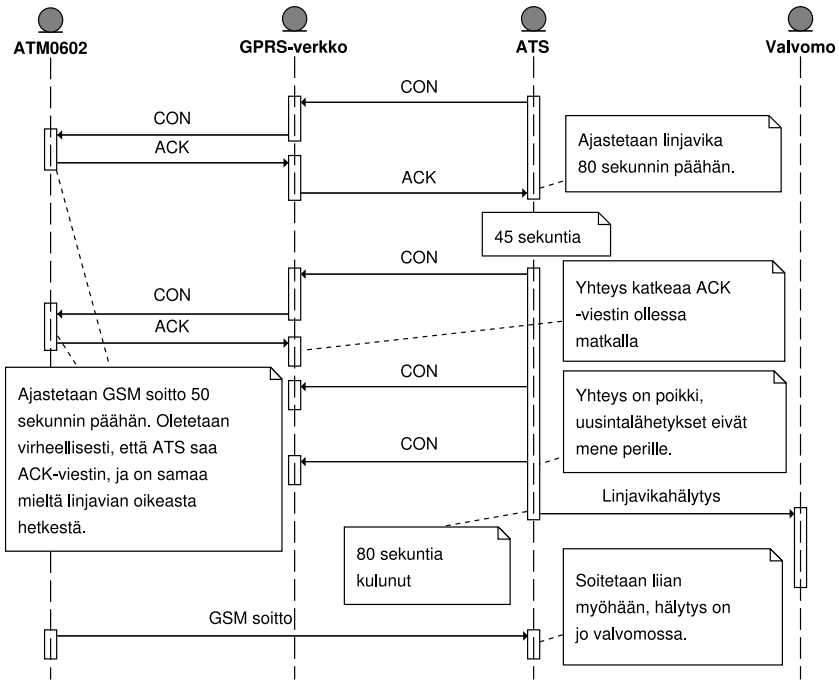
Jos liikennöintitehtävä jollain silmukkansa suorituskerralla havaitsee ensiksi mainitun lippubitin olevan asetettuna, se alkaa avata yhteyttä. Onnistuessaan se vaihtaa bitin arvon takaisin, tai sitten epäonnistuessaan se asettaa *TCP/IP-yhteyden avaaminen epäonnistui* -bitin. Kummassakin tapauksessa ohjelmalogiikka pääsee suoritukseen. Se asettaa molempien bittien arvot nolliksi, ja sitten joko siirtyy seuraavaan tilaan tai yrittää uudelleen, riippuen kumman bitin arvon vaihtumisen vuoksi se herätettiin. Tietenkin tässä vaiheessa saatetaan myös havaita, että on aika soittaa ATS:lle kontrollisoitto. Jos näin on tehty, palataan aina tilakoneen ensimmäiseen tilaan siitä syystä, että jos kyseessä oli langaton pääyhteys, soiton tekeminen katkaisee aina internet-yhteyden. Tämä on käytetyn GSM-moduulin ominaisuus.

5.9 Langattoman toiminnan toteuttaminen

GPRS-yhteyttä kokeiltiin ensimmäisen kerran jo vuoden 2006 lokakuussa. Silloin käytettiin kehitysympäristön omaa PPP-ohjelmakirjastoa, joka oli jo muutenkin mukana ohjelmakoodissa käyttöliittymän sarjakaapelilla käyttämistä varten. Se ei kuitenkaan osoittautunut riittävän luotettavaksi, vaan testeissä esiintyi yhteyden selittämätöntä katkeilua sekä resettejä. Yhteyden toteuttaminen jätettiin silloin myöhemmäksi.

GPRS-yhteyden toteuttamiseen palattiin vuoden 2007 elokuun lopussa. Tällä kertaa päätettiin tehdä kokonaan oma, AT-komentoihin perustuva toteutus. Tehtävien jako tekijöiden välillä oli sama kuin aiemminkin. Kirjoittajalla oli tämän ominaisuuden suhteen aiempaa vähemmän tekemistä. Joitain aikarajoja oli pidennettävä ja lisättävä erilaisia varmistuksia kontrolli- ja datasoittoihin. Esimerkiksi estettiin se, että kontrollisoitto tehtiin juuri ennenkuin GPRS olisi nopeimmillaankaan ehtinyt muodostaa internet-yhteyden.

Jos yhteyden muodostamista aloittaessa havaittiin, ettei se voisi mitenkään ehtiä, tehtiin kontrollisoitto välittömästi ja sitten aloitettiin jälleen internet-yhteyden muodostus, kun sen suorittamiseen oli nyt enemmän aikaa. Ensimmäinen asiakkaiden käyttöön annettu versio, jossa oli virallinen GPRS-tuki, julkaistiin marraskuussa. Aluksi GPRS-yhteys rajoitettiin UDP:hen. Vasta lähes vuoden kuluttua julkaistiin ensimmäinen versio, jossa oli luotettavasti toimiva TCP, kun käytössä oli GPRS.

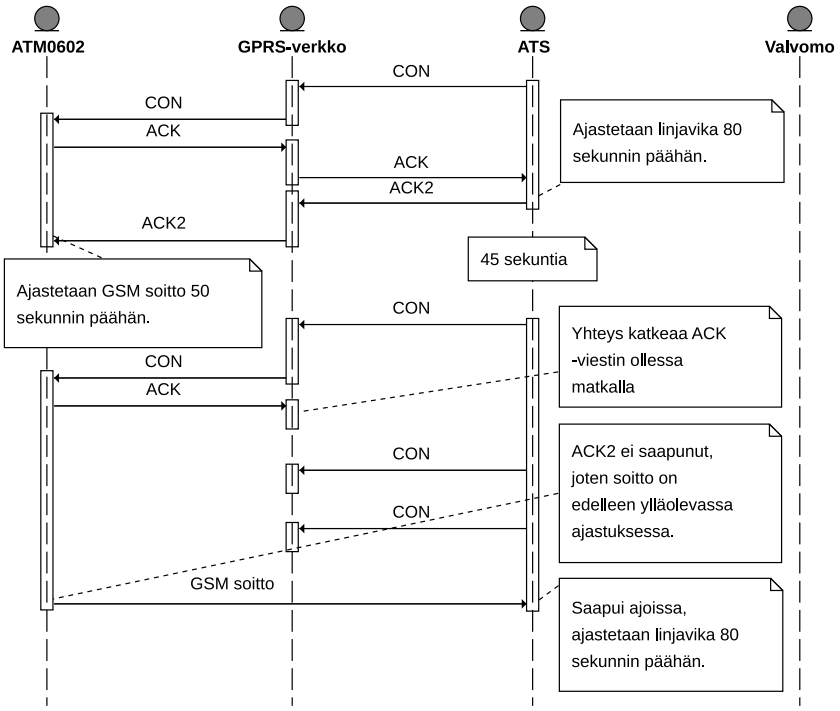


Kuva 5.9: *Tapahtumasekvenssikaavio virheellisestä linjavikahälytyksestä, kun käytetään hidasta internet-yhteyttä ja yhteys katkeaa tietyllä hetkellä*

5.10 Projektin ylläpitovaihe

Enemmän kirjoittajan aikaa vei Telcont Oy:n kehittäessä GPRS-yhteyttä, uusi settings-viesteihin liittyvä ominaisuus. Asetukset haluttiin nyt olevan muutettavissa myös ATS:lle tehdystä, uudesta käyttöliittymäsivusta. Settings-viestit välitettäisiin ATM0602:lle samaan tapaan kuin palautettavat, vanhemmat asetukset. Tähän liittyen havaittiin kaikenlaisia pieniä ongelmia, kuten että ATM0602:lle ilmoitettua ATS:n puhelinnumeroa vaihtaessa ohjelma ei palannut alkutilaansa, jolloin myös GSM-ajuri olisi ollut tietoinen numeron muuttumisesta. Tai yhteyskäytäntöä vaihtaessa UDP:n ja TCP:n välillä ATS:n kautta, saattoi mennä jotain vikaan, ja nyt laitteeseen ei saanutkaan enää yhteyttä kuin menemällä sarjakaapelin kanssa paikalle.

Tässä vaiheessa laitteita oli myyty jo satoja, joten nämä olivat todellisia ongelmia vaikka niiden ratkaisut olivatkin yksinkertaisia. Suurempi ongelma oli saada tarkkaa tietoa siitä, mitä jossain toisessa kaupungissa olevassa laitteessa oikein oli tapahtunut ja sitten toistaa tilanne omassa laitteessa. Yksi esimerkki tällaisesta ongelmasta näkyy kuvassa 5.9, jossa ACK-viesti katoaa matkalla, pahimmalla mahdollisella hetkellä. Ainut ulkoinen oire tämän tapahtumisesta on lyhyt linjavika, ja syy tähän havaittiin täysin sattumalta, aivan toista ominaisuutta testattaessa.



Kuva 5.10: *Tapahtumasekvenssikaavio virheellisen linjavikahälytyksen välttämisestä ACK2-viestin avulla.*

ACK-viestin katoaminen on tietenkin ollut alusta asti teoriassa mahdollista, mutta langallisella internet-yhteydellä viiveajat ovat niin lyhyitä, että yhteyskatkon tapahtuminen juuri kuvatussa kohdassa oli erittäin epätodennäköistä. GPRS sen sijaan pidensi viiveaikoja, ja nyt ongelma alkoi jo silloin tällöin realisoitua. Tilanne lähtee liikkeelle siitä, että yhteys katkeaa hieman sen jälkeen, kun ATM0602 on lähettänyt vahvistuksen kontrollisanomaan. ATM0602:n näkökulmasta yhteys on siis kunnossa, koska kontrollisanoma saapui, kun taas ATS katsoo yhteyden katkenneen, koska vahvistusta ei tullut. ATM0602 ei siten tee kontrollisoittoa silloin kun ATS sitä odottaa, ja seurauksena on linjavika. Kun sitten ATM0602 lopulta havaitsee, että seuraavaa kontrollisanomaa ei tulekaan, se tekee kontrollisoiton. Tällöin se on jo liian myöhäistä. Linjavialta paluu tapahtuu, kun ATM0602 rekisteröityy uudelleen, tehtyään kontrollisoiton ja saatuaan ATS:ltä kiellon soittaa uudelleen, koska linjavikahälytys oli jo lähetetty.

Ongelma korjattiin kuvan 5.10 mukaisella tavalla. Kaikkiin ATM0602:n lähettämiin vahvistussanomiin odotetaan saatavan vahvistuksen vahvistus, eli ACK2-viesti ATS:ltä. Ylimääräistä viestiä tarvitaan vain näin päin liikennöitäessä. Jos alkuperäisen viestin lähetti ATM0602, vastaava ongelmatilanne ei ole mahdollinen. Kuten kuvasta ilmenee, jos yhteys nyt katkeaisi samal-

la tavalla, ATM0602 havaitsisi sen siitä, ettei se saisi ACK2-viestiä, ja tekisi kontrollisoiton. Tällöin linjavikaa ei tapahtuisi.

Yksi kriittinenkin virhe havaittiin ylläpitovaiheessa. Erään viestin otsikkokentät määrittelevä tietorakenne ei vastannut täysin todellista viestiä, mikä johti siihen, että sen pituus laskettiin tapahtumapuskuriin sijoitettaessa väärin. Tämä olisi voinut aiheuttaa määrittelemätöntä toimintaa jos tapahtumat päätyvät sopivassa järjestyksessä puskuriin. Testeissä tilannetta ei saatu aikaan, mutta koska virhe oli kuitenkin osoitettu teoriassa, ja riski siitä, että pahimmassa tapauksessa hälytys jää välittämättä on äärimmäisen vakava, toimitettiin asiakkaiden laitteisiin pikaisesti päivitys. Yhdestäkään tapauksesta, jossa riski olisi ehtinyt toteutua, ei olla kuultu.

6. YHTEENVETO

Projekti saavutti kaikki sille asetetut tavoitteet. Yksikään tapaus, jossa hälytys olisi jäänyt saapumatta perille ohjelmavirheen vuoksi, ei ole tullut Telcont Oy:n tietoon. ATM0602-moduulia käyttävistä vartiointipalveluista mainittakoon OTSO vartiointi Oy sekä Hämeen Lukko. Ensiksi mainittu käyttää ratkaisusta nimeä *VARMAverkko* ja jälkimmäinen taas nimitystä *IsmoD4*, josta on kirjoitushetkellä ilmoitus heidän verkkosivunsa (*www.hameenlukko.fi*) etusivulla.

Osa laitteista on tosin myöhemmin toteutettua *MTM*-mallia, jonka toteutus jää tämän diplomityön aihepiirin ulkopuolelle. Siinä on ATM0602:sta kopioitu logiikkaprosessi, ja sen ohjelmistoarkkitehtuurista noin kolmasosa on kirjoittajan työtä. ATM0602:n ohjelmistoarkkitehtuuri on erittäin onnistunut. Se on pysynyt samanlaisena ensimmäisestä, pelkkää UDP:tä langallisella yhteydellä tukevasta versiosta asti, eikä mitään sen jälkeen toteutettua ominaisuutta tehdessä ole tarvinut kiertää sen aiheuttamia ongelmia.

Kaikki erityistä suunnitelmallisuutta vaatinut ohjelmistokehitys tapahtui vuoden 2006 kesällä. Ylivoimaisesti vaikein osa projektissa oli käyttöliittymän toteuttaminen CGI-rajapinnan avulla. Jos jossain vaiheessa tarvitsee toteuttaa toinen, käyttöliittymän sisältävä laite samalla laitealustalla, on ATM0602:n käyttöliittymä arvokas ja helposti siirrettävä komponentti. Projektin yhteydessä kehitetty merkintäkieli soveltuu monenlaiseen käyttöön. Kirjoittajan suunnittelema ulkoasu on jo kopioitu ATS:ään sekä parhaillaan kehitteillä olevaan, ATS:n uudistettuun versioon, *DASP*:iin (Duplicated Alarm Supervising Proxy), jonka tärkein ominaisuus on, että se on laitteistoltaan kahdennettu. Jos toinen laite vikaantuu, DASP siirtyy automaattisesti käyttämään toista laitetta muutaman kymmenen sekunnin viiveellä.

Rabbit Semiconductorin tuotteiden käyttämistä ei kirjoittaja voi varauksetta suositella Dynamic C:n puutteiden vuoksi. Se poikkeaa ANSI-C-standardista niin paljon, että muiden ohjelmien, kuten Emacs-editorin käyttäminen on työlästä. Lisäksi projektin aikana on jäänyt se vaikutelma, että vakavatkin virheet kehitysympäristössä korjataan hitaasti. Viimeisimmässäkin versiossa on yhä korjaamatta DHCP-kirjastossa oleva virhe, jonka vuoksi IP-osoite jää uudistamatta, jos DHCP-palvelimelta saapuu ilmoitus istunnon

vanhenemisesta. Korjaus vaatisi vain kaksi ohjelmakoodiriviä kirjastoon, ja se on ollut internet-foorumeilla esitettynä jo yli vuoden ajan. Onneksi mukana tulevat kirjastot toimitetaan lähdekoodeineen, joten muutoksen voi tehdä itse. Doxygen-ohjelmiston käyttäminen on sen sijaan erittäin suositeltavaa.

Asia, joka projektissa on käynyt useaan kertaan ilmi on, että kahden laitteen välisen yhteyden toteuttaminen internetin yli on aikaa vievää. Ongelmia ei saada esiin tekemällä kehitystä tuotekehitysyksikön lähiverkossa. Vasta kenttätestit paljastavat, mitä kaikkia yllätyksiä voi seurata epästandardinmukaisista kytkimistä, palomuureista ja DHCP-palvelimista, internetpalveluntarjoajien tekemistä yöllisistä päivityksistä ja vastaavista todellisen käyttöympäristön mukanaan tuomista ongelmista. Laitteet kannattaa jo valmiiksi ohjelmoida tekemään runsaasti lokia internetiin liittyvistä toimistaan niin, että ongelmien toteutuessa on mahdollista päästä niistä jäljille. Verkkoliikenteen tarkkaileminen esimerkiksi Wireshark-nimisellä ilmaisohjelmalla on myös välttämätön osa tuotekehitystä. On varsin kustannustehokasta lähettää ainakin muutama työntekijä verkkotekniikkoja käsittelevälle kurssille oppimaan Wiresharkin näyttämän liikenteen tulkitsemista, kuten Telcont Oy:ssa toimittiin.

LÄHTEET

[1] 019-0108T. *Rabbit 3000 Microprocessor User's Manual*. USA, 2009. Digi International Inc. 302 p.

[2] 019-0136. 080930-K. *RabbitCore RCM3700 User's Manual*. USA, 2008. Digi International Inc. 169 p.

[3] 090409-M. 019-0113. *Dynamic C Function Reference Manual*. USA, 2008. Digi International Inc. 560 p.

[4] DAVIDOW, A. Computer screens are not like paper: typography on the web. In: Sassoon, R. (ed.). *Computers and Typography 2*, first ed. Intellect Books, Wiltshire, Great Britain, 2002, pp. 21 – 40.

[5] FOSTER, J., OSIPOV, V. & BHALLA, N. *Buffer Overflow Attacks: Detect, Exploit, Prevent*, first ed. Syngress Publishing, Rockland, Massachusetts, USA, 2005. 512 p.

[6] HAIKALA, I. JA JÄRVINEN, H.-M. *Käyttöjärjestelmät*. Talentum Media Oy, Jyväskylä, 2003. 242 p.

[7] HAKULINEN, J. *Tiivistelmä patenttihakemuksista FI20051025, FI20051148 ja FI20060050*. Telcont Oy, Kouvola, 2005. 17 p.

[8] HAKULINEN, J. *Ilmoituksensiirtojärjestelmien historia -esityksen kalvot*. Kouvola, 2008. Telcont Oy. Julkaisematon selvitys. 27 kalvoa.

[9] HASHIMOTO, A. *Visual Design Fundamentals: A Digital Approach*, first ed. Charles River Media, Hingham, Massachusetts, USA, 2003. 345 p.

[10] HEINE, G. & SAGKOB, H. *GPRS: Gateway to Third Generation Mobile Networks*, second ed. Arctech House, Inc., Norwood, Massachusetts, USA, 2003. 328 p.

[11] KOPETZ, H. *Real-Time Systems: Design Principles for Distributed Embedded Applications*, first ed. Kluwer Academic Publishers, Boston, USA, 1997. 352 p.

[12] LABROSSE, J. *MicroC/OS-II: The Real Time Kernel*, second ed. CMP Books, San Francisco, California, USA, 2002. 648 p.

[13] MCCLURE, S., SCAMBRAY, J. & KURTZ, G. *Hacking Exposed*, fifth ed. McGraw-Hill/Osborne, Emeryville, California, USA, 2005. 692 p.

[14] MELAKOSKI-VISTBACKA, S. *Käyttöliittymäsuunnittelun Perusteet, luentomoniste*, 1.0 ed. Tampereen teknillinen yliopisto, Hervanta, 2006. 53 p.

[15] RFC5735. *TCP SYN Flooding*, 2007. Internet Engineering Task Force. 19 p.

[16] RFC5735. *Special Use IPv4 Addresses*, 2010. Internet Engineering Task Force. 11 p.

[17] SCHUH, P. *Integrating Agile Development in the Real World*, first ed. Charles River Media, Hingham, Massachusetts, USA, 2004. 364 p.

[18] SFS-EN 50136-1-1 + A1 + A2. *Hälytysjärjestelmät. Ilmoituksensiirtojärjestelmät ja laitteet. = Alarm systems. Alarm transmission systems and equipment.* Helsinki, 2008. Suomen standardoimisliitto. 36 p.

[19] STALLINGS, W. *Data & Computer Communications*, sixth ed. Prentice Hall, Upper Saddle River, New Jersey, USA, 2000. 810 p.

[20] THURSTON, S. *Managing Cisco Network Security*, second ed. Syngress Publishing, Rockland, Massachusetts, USA, 2002. 752 p.

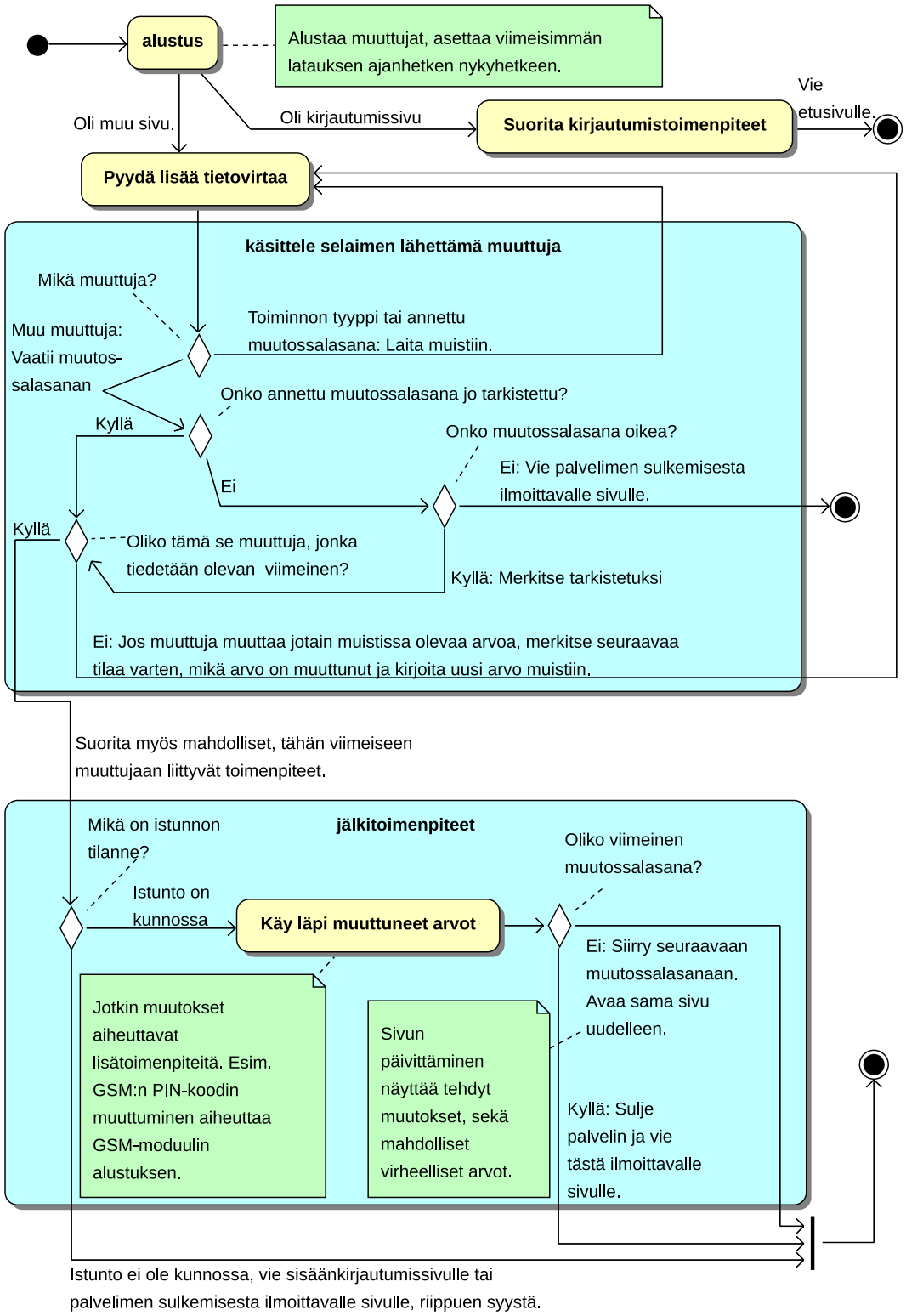
[21] VIEGA, J. & MESSIER, M. *Secure Programming Cookbook for C and C++ (Safari Online eBook)*, first ed. O'Reilly Media, Inc., Sebastopol, California, USA, 2003. 792 p.

LIITE A: KÄYTTÖLIITTYMÄN PARAMETRIT

```
typedef struct {
   /// Input method for parameter (text field, dropdown etc.)
   unsigned char form_type;
   /// Data type. Acceptable types are macros such as UINT16_TYPE
   /// or INET_STR_TYPE.
   unsigned char type;
   /// Parameter's internal name: The name in HTML fields.
   char* internal_name;
   /// Parameter's name as displayed to user.
   char* name_str;
   /// Pointer to parameter data.
   void* pVar;
   /// Pointer to default value of parameter data.
   void* pVarDefault;
   /// If dropdown, how many items in it. Otherwise 0.
   unsigned char items;
   /// Dropdown item strings and their corresponding values. NULL
   /// if not dropdown.
   parameter_select_t* pItem_list;
   /// Minimum value if number, or minimum length if string.
   unsigned long min;
   /// Maximum value if number, or maximum length if string.
   unsigned long max;
   /// Possible special operations that must be performed if
   /// value changed
   unsigned char special_function;
} parameter_t, *pParameter_t;

const parameter_t parameterList[] = {

   {FORM_TYPE_SELECT, UINT16_TYPE, "DHCP", "DHCP", &conf.DHCP,
   &confDefault.DHCP, sizeof(onSelect) / sizeof(parameter_select_t),
   &onSelect[0], 0, 1, SPECIAL_NONE},

   {FORM_TYPE_TEXT, INET_STR_TYPE, "IP", "IP", &conf.IPAddr,
   &confDefault.IPAddr, 0, NULL, 0, 0, SPECIAL_DHCP_IP},

   {FORM_TYPE_TEXT, INET_STR_TYPE, "netm", "Netmask", &conf.netmask,
   &confDefault.netmask, 0, NULL, 0, 0, SPECIAL_DHCP_NETMASK},

   {FORM_TYPE_TEXT, INET_STR_TYPE, "gw", "Gateway", &conf.gateway,
   &confDefault.gateway, 0, NULL, 0, 0, SPECIAL_DHCP_GATEWAY},
```

Ohjelma A.1: Osa käyttöliittymän yleisten asetusten sivun parametrilistaa, jonka perusteella sivun HTML-sisältö automaattisesti muodostetaan.

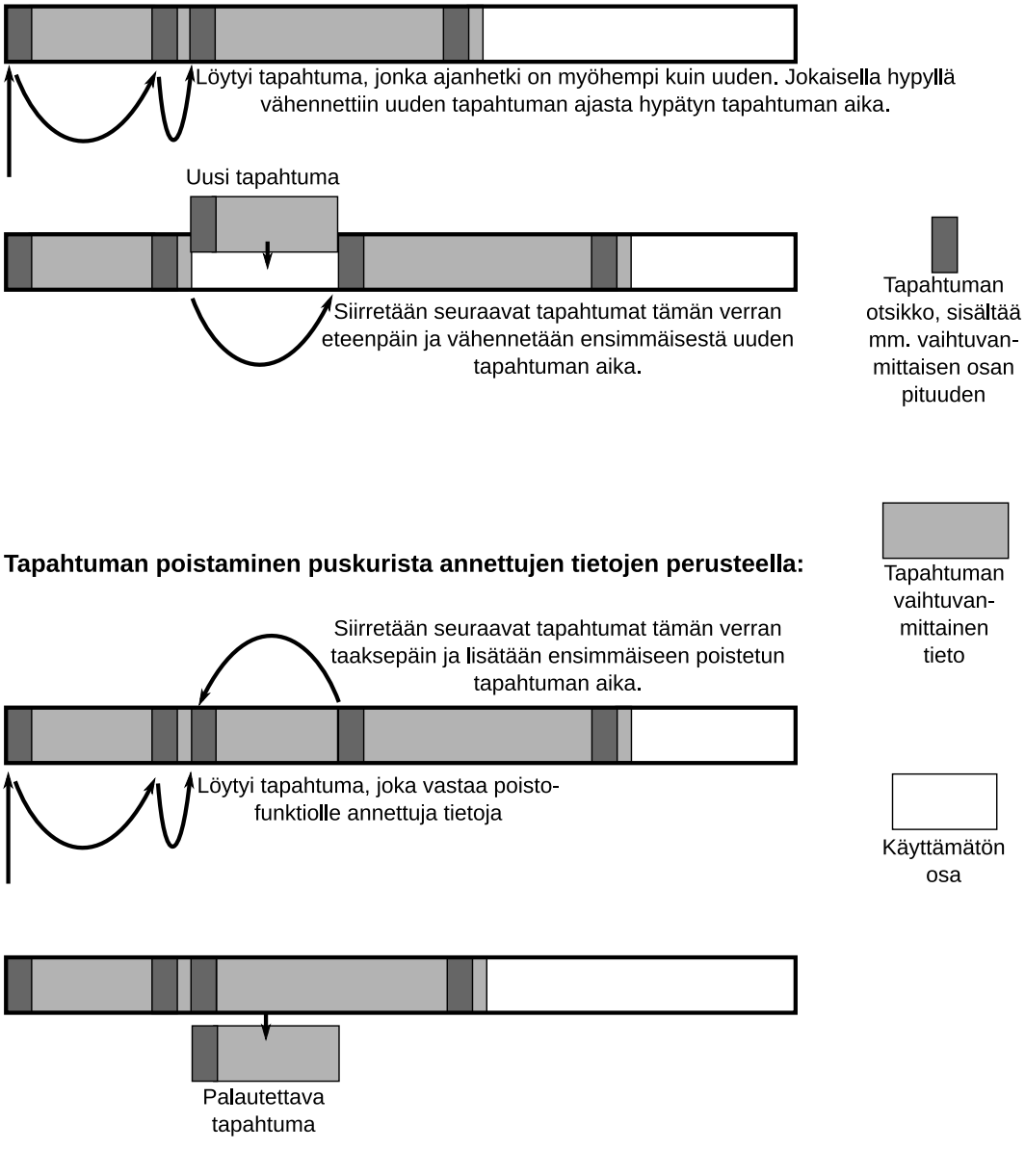
LIITE B: SUBMIT-FUNKTIOTA KUVAAVA TILAKONE



Kuva B.1: *Tilakone, joka kuvaa submit-funktion toimintaa*

LIITE C: TAPAHTUMAPUSKURIN TOIMINTA

Uuden tapahtuman sijoittaminen puskuriin:



Tapahtuman poistaminen puskurista annettujen tietojen perusteella:

Kuva C.1: *Tilakone, joka kuvaa submit-funktion toimintaa*