LAPPEENRANNAN TEKNILLINEN YLIOPISTO

TEKNISTALOUDELLINEN TIEDEKUNTA

TIETOTEKNIIKAN OSASTO

TIEDON HAJAUTUS JOHTAMISJÄRJESTELMÄSSÄ JA DDS-TEKNOLOGIA

Diplomityön aihe on hyväksytty Lappeenrannan teknillisen korkeakoulun tietotekniikan osaston osastoneuvoston kokouksessa 3.3.2010.

Työn tarkastajina toimivat dosentti, TkT Jouni Ikonen ja DI Markku Tienhaara.

TIIVISTELMÄ

Lappeenrannan teknillinen yliopisto
Teknistaloudellinen tiedekunta
Tietotekniikan osasto
Teemu Salmivesi

Tiedon hajautus johtamisjärjestelmässä ja DDS-teknologia

Diplomityö
2010
61 sivua, 18 kuvaa, 1 taulukko ja 2 liitettä

Tarkastajat: Dosentti, TkT Jouni Ikonen
DI Markku Tienhaara

Hakusanat: Johtamisjärjestelmä, verkostokeskeinen sodankäynti, välikerrosarkkitehtuuri, Data Distribution Service, OpenSplice, palvelun laatu

Keywords: Command and control system, network-centric warfare, middleware architecture, Data Distribution Service, OpenSplice, Quality of Service

Verkostokeskeisessä sodankäynnissä tietojärjestelmien suurimpana haasteena on oikean tiedon hajauttaminen oikeaan paikkaan ja aikaan. Tietojärjestelmissä esitettävän ilmatilannekuvan tulee vastata reaalimaailman tilannetta parhaalla mahdollisella tavalla. Ilmatorjunnassa reaaliaikaisuus nousee erityisen suureen rooliin nopeasti liikkuvien kohteiden takia.

Tämä diplomityö on tehty Insta DefSec Oy:ssä liittyen johtamisjärjestelmän uudistamishankkeeseen. Työn vaatimuksina olivat standardeihin perustuvat ratkaisut, joista keskeisimmäksi nousi Data Distribution Service -standardi (DDS) ja sen hyödyntäminen osana johtamisjärjestelmän tiedon hajautusta. Työssä esitellään johtamisjärjestelmien tiedon hajautukseen liittyviä haasteita sekä paikallisessa että maantieteellisesti hajautetussa toimintaympäristössä.

Työssä toteutettiin liityntäohjelmisto nykyisen ja uuden johtamisjärjestelmän välille. Liityntäohjelmiston tehtävänä on tuottaa reaaliaikaista ilmatilannekuvaa nykyisestä johtamisjärjestelmästä uuteen johtamisjärjestelmään. DDS-standardin toteuttavana välikerrosarkkitehtuurina käytettiin OpenSplice DDS -tuotetta. Valittu teknologia tarjoaa edistykselliset julkaisija–tilaaja-mallin mukaiset menetelmät tiedon reaaliaikaiseen hajauttamiseen. DDS:n arkkitehtuuri ja palvelun laadun mekanismit mahdollistavat tiedon hajautuksen sodanajan johtamisjärjestelmille.

ABSTRACT

Lappeenranta University of Technology
Faculty of Technology Management
Department of Information Technology
Teemu Salmivesi

DDS for Data Distribution of the Command and Control System

Thesis for the Degree of Master of Science in Technology
2010
61 pages, 18 figures, 1 table and 2 appendices

Examiners: Adjunct professor, DSc. Jouni Ikonen
MSc. Markku Tienhaara

Keywords: Command and control system, network-centric warfare, middleware architecture, Data Distribution Service, OpenSplice, Quality of Service

In systems of network-centric warfare, there are common characteristics, such as data, that must be distributed to the right place at the right time. The displayed Air Situation Picture (ASP) on command and control systems has to meet the real world snapshot in the best possible way. Because of fast moving targets, real-time actions have a particularly significant role in air defense systems.

This thesis was made in Insta DefSec Ltd as a part of command and control system modernization program. The task was to acquire solutions based on specific standards. The key standard for this thesis was the Data Distribution Service (DDS). The main objective was to use DDS as a part of data distribution solution in the command and control system. This thesis will focus on challenges of data distribution in both centralized and decentralized manner including mobile command centers.

In this thesis the application interface solution was defined, planned and implemented so as to distribute air situation picture from the current command and control system to the new command and control system. The OpenSplice DDS product was chosen as a middleware layer to fulfill standard requirements of DDS. The selected technology will provide advanced publish-subscribe model for real-time information in accordance with the methods of data distribution. DDS's architecture and the quality of service mechanisms enable the data distribution of command and control systems in times of war.

ALKUSANAT

Tämä työ on tehty Insta DefSec Oy:lle Tampereella. Työn ohjaajana toimivat dosentti, TkT Jouni Ikonen Lappeenrannan teknillisestä yliopistosta sekä DI Markku Tienhaara Insta DefSec:sta. Kiitän heitä kannustavasta ja rakentavasta palautteesta diplomityön kirjoittamiseksi siitäkin huolimatta tai juuri sen takia, että opintoja on vuosissa ehtinyt kertyä vähintään kunnioitettava määrä. Haluan myös kiittää työnantajaani tuesta ja mahdollisuudesta tehdä tämä diplomityö.

Samoin haluan kiittää perhettäni pitkäpinnaisuudesta opintojeni suhteen sekä vanhempiani, jotka opintojen kuluessa ovat tukeneet ja kannustaneet minua.

Tampere, Maaliskuu 2010

Teemu Salmivesi

SISÄLLYSLUETTELO

TERMIT JA LYHENTEET

ADCC	Air Defence Command & Control on ilmapuolustuksen johtamisjärjestelmä.
BOOST	Boost on ilmainen avoimeen lähdekoodiin perustuva alustariippumaton C++-kielellä toteutettu luokkakirjasto. Kirjasto tarjoaa mm. muistinhallinnan ja säikeistyksen mekanismeja.
CORBA	The Common Object Request Broker Architecture.
DCPS	Data-Centric Publish-Subscribe, OMG:n standardin mukainen rajapinta.
DDS	Data Distribution Service, reaaliaikajärjestelmissä käytetty tietoliikenneväylä, joka perustuu julkaisija–tilaaja-malliin.
DLRL	Data Local Reconstruction Layer, OMG:n standardin mukainen rajapinta.
Etyj	Euroopan turvallisuus- ja yhteistyöjärjestö.
GCC	the GNU Compiler Collection, GNU lisenssin alainen C/C++ kääntäjä
GDS	Global Data Space, globaali tietoavaruus DDS standardissa.
IDL	Interface Definition Language, rajapintojen kuvauskieli, jota käytetään CORBA- ja DDS-arkkitehtuurissa.
JMS	Java Message Service, Sun Microsystemsin kehittämä viestinvälitysstandardi.
LAN	Local Area Network eli lähiverkko, joka on rajoitetulla alueella toimiva tiedonsiirtoverkko.

LGPL	GNU Lesser General Public License, GNU organisaation määrittelemä lisenssi, jonka alaisia tuotteita voidaan tietyin ehdoin käyttää ilmaiseksi osana kaupallista tuotetta.
MRT	Multi Radar Tracking on monitutkaseuranta laskentajärjestelmä, jonka avulla luodaan reaaliaikaista ilmatilannekuvaa koko valtakunnan alueelle.
MST	Multi Sensor Tracking on monisensorijärjestelmä, jonka avulla luodaan reaaliaikaista ilmatilannekuvaa koko valtakunnan alueelle.
NATO	the North Atlantic Treaty Organisation, Pohjois-Atlantin liitto on kansainvälinen puolustusliitto.
OMG	Object Management Group on voittoa tavoittelematon konsortio, joka tuottaa standardeja hajautettuihin järjestelmiin.
PTP	Point-to-Point, JMS standardin tukema välitystapa, jossa viestit menevät jonossa lähettäjältä vastaanottajalle.
QoS	Quality of Service, tietoliikenteen resurssien kontrolloinnin mahdollistava mekanismi.
RWS	Radar WorkStation on järjestelmä, joka on osana tutka-asemaa.
SIMD	SIMple DDS, yksinkertaisempi C++-API DCPS-rajapinnan toteuttamiseen.
TDT	Tactical Data Terminals on päätejärjestelmä, joka mahdollistaa liittymisen kiinteisiin tiedonsiirtoverkkoihin radioverkoista.
TTL	Time to live on tieto jolla määritetään IP-pakettien elinkaari verkossa. TTL on osa IP-protokollan otsikkotietoa.
URI	Uniform Resource Identifier on nimen tai resurssin Internetissä identifioiva teksti.

VLAN	Virtual local area network on virtuaalinen lähiverkon palveluita tarjoava teknologia WAN-tiedonsiirtoverkossa.
WAN	Wide Area Network on tiedonsiirtoverkko, joka peittää laajoja maantieteellisiä alueita. Diplomityössä tällä tarkoitetaan Suomen kattavaa tiedonsiirtoverkkoa.
XML	eXtensible Markup Language on merkintäkieli, jolla tiedon merkitys kuvataan tiedon sekaan. Kuvauskieli auttaa jäsentämään tietoa selkeämmin.
YK	Yhdistyneet Kansakunnat.

1. JOHDANTO

”Ilmavoimat vastaa maamme ilmatilan jatkuvasta valvonnasta ja vartioinnista. Ilmatilan loukkaukset estetään tarvittaessa voimakeinoin. Sodan aikana ilmavoimien päätehtävä on hävittäjätorjunta. Ilmavalvonta antaa edellytykset riittävään tilanteen hallintaan, jotta käytettävissä oleva torjuntavoima voidaan keskittää oikeaan aikaan oikeaan paikkaan komentajan käskemällä tavalla. Oikea-aikainen reagointi edellyttää kykyä muodostaa kattava tilannekuva toimintaympäristöstämme niin rauhan kuin mahdollisen kriisinkin aikana.” (Ilmavoimat, 2009)

Ilmatilannekuvan avulla pyritään luomaan johtamisjärjestelmiin reaaliaikainen tilanne Suomen ja sen raja-alueiden ilmatilasta. Ilmatilannekuva muodostuu useista erilaisista toisiaan täydentävistä sensoreista, joiden tietoa analysoimalla saadaan luotua luotettava, ymmärrettävä ja yksikäsitteinen ilmatilannekuva. Ilmatilannekuvaa tuottavia sensoreita ovat tällä hetkellä pääasiassa lähi- ja kaukovalvonnan tutkat, joita on noin 20 kappaletta eri puolilla Suomea (Ruotuväki, 2006). Ilmatilannekuvaa täydennetään hävittäjien tutkien tuottamilla havainnoilla ja maastossa olevien havaintoasemien eli aisti-ilmavalvontapaikkojen tuottamilla tiedoilla. Rauhanaikana Ilmavoimien tehtävänä on Suomen ilmatilan valvonta. Rauhanajan ilmatilannekuvaa täydennetään ilmailulaitoksen tuottamilla lentosuunnitelmatiedoilla, joiden perusteella voidaan tunnistaa jokainen yksittäinen Suomen ilmatilassa lentävä lentokone (Puolustusvoimat, 2002).

Julkisuudessa on ollut esillä Suomen ilmatilaan kohdistuneita loukkauksia, jotka valvonta- ja johtamisjärjestelmät ovat pystyneet luotettavasti ja todistettavasti havaitsemaan. Ilmatilan loukkaus on vieraan maan lentokoneen luvaton lentäminen Suomen ilmatilassa, joka on Suomen valtion maa- ja merialueen yläpuolella sijaitseva alue. Ilmatilanloukkauksia pidetään vakavina tapahtumina ja

niitä on julkisuudessa käsitellyt näyttävästi esimerkiksi Yleisradio vuonna 2008 (Kuva 1).

Kuva 1 Suomen ilmatilan loukkaus (Yleisradio, 2008)

1.1 Taustaa

Diplomityön vaatimuksiin ja aihepiirin tutkimukseen merkittävästi vaikuttava asia on valtioneuvoston uusi puolustuspoliittinen selonteko. Selonteossa tuodaan esille Puolustusvoimien tarve panostaa entistä enemmän Suomen ulkoisen toiminnan kehittämiseen. Ulkoisen toiminnan kehittämisellä tarkoitetaan tässä yhteydessä Pohjoismaiden, Euroopan unionin jäsenmaiden, YK:n, Etyjin ja puolustusliitto Naton kanssa tapahtuvaa yhteistyötä ja yhteistoimintakyvyn kehittämistä. Tietojärjestelmien kannalta tämä tarkoittaa yhteistoiminnassa olevien

kumppaneiden tietojärjestelmien yhteensopivuutta Suomen puolustusvoimien tietojärjestelmiin (Valtioneuvoston selonteko, 2009). Järjestelmien yhteensopivuuden lähtökohtana ovat standardeihin pohjautuvat ratkaisut, joiden avulla pystytään helpommin ja kattavammin vastaamaan yhteistoiminnan vaatimuksiin.

Ilmatilannekuvan lähteinä toimivat sensorit ovat myös uudistumassa. Suomi on siirtymässä Multi Radar Tracking (MRT) järjestelmästä Multi Sensor Tracking (MST) monisensorijärjestelmään. MRT-järjestelmässä ilmatilannekuva tuotetaan pääasiassa lähi- ja kaukovalvontatutkilla, mutta saksalaiselta EADS Deutschland GmbH:lta hankittavaan MST-järjestelmään voidaan syöttää useasta eri lähteestä tulevia tietoja, joilla voi olla merkitystä johtamisen perustana käytetyn maalitilannekuvan muodostamisessa (Ilmavoimat, 2004).

Järjestelmät tulevat kehittymään entistä enemmän tietofuusiojärjestelmiksi, jotka samalla tukevat valvontaan ja torjuntaan liittyvää päätöksentekoa. Reaaliaikainen tilannekuva verkottuu edelleen ja siihen tulevat integroitumaan sekä kaikkien puolustushaarojen tiedustelu- ja valvontajärjestelmät että taisteluyksiköiden oma tilannekuva. Enää ei siis puhuta pelkästään ilmatilannekuvasta vaan kaikkien puolustushaarojen joukkoja koskevasta yhteisestä tilannekuvasta. Johtamisjärjestelmät ovat muuttumassa nykyaikaisiksi verkostokeskeisiksi johtamisjärjestelmiksi. Verkostokeskeisissä johtamisjärjestelmissä tiedon määrän lisääntyminen edellyttää suurta tiedonsiirtokykyä. Liikkuvia yksiköitä varten tarvitaan automaattisesti mukautuvaa viestiverkkoa. Järjestelmiin tulee pystyä liittämään ja tarvittaessa irrottamaan siirrettäviä, nopeasti liikkuvia joukkoyksiköitä. Järjestelmiin tulee pystyä liittämään myös kansainvälisten yhteistoimintakumppaneiden tietojärjestelmiä (Jussila, 2004). Yhteistoimintakyvyn kehittäminen ja uusi MST-järjestelmä tulevat aiheuttamaan muutostarpeita nykyisin käytössä oleviin johtamisjärjestelmiin, mutta ne tulevat myös olemaan lähtökohtana suunniteltaessa tulevaisuuden johtamisjärjestelmiä.

1.2 Tavoitteet ja rajaukset

Diplomityön tarkoitus on tutkia, suunnitella ja toteuttaa tulevaisuuden ilmapuolustuksen johtamisjärjestelmän tiedon hajautusta. Tutkimuksessa keskitytään hajautuksessa käytettävään teknologiaan, jonka ominaisuuksia analysoidaan suhteessa johtamisjärjestelmän asettamiin tiedon hajautuksen vaatimuksiin. Suunnittelussa ja toteutuksessa keskitytään johtokeskuksen sisällä olevien järjestelmien ja operaattoreiden väliseen tiedon hajauttamiseen sekä hahmottelemaan hajautettujen johtokeskusten nopeasti muuttuvien tietojen hajautusta.

Diplomityö tehdään osana suurempaa hanketta, jonka tavoitteena on uuden johtamisjärjestelmän toteuttaminen Puolustusvoimille. Projektissa toteutettavien ratkaisujen tulee perustua yleisesti tunnettuihin komponentteihin ja standardeihin. Projektin tavoitteena on tuottaa ratkaisut tehokkaasti ja innovatiivisesti, käyttäen avoimen lähdekoodin ratkaisuja ja niitä tukevia avoimia standardeja.

1.3 Työn rakenne

Johdantoa seuraavassa luvussa käsitellään toimintaympäristöä, johon tiedonhajautuksen ratkaisua ollaan suunnittelemassa. Siinä esitellään johtamisjärjestelmien ominaispiirteitä ja diplomityön kannalta keskeisempiä ongelmakenttiä tiedon hajautuksen osalta. Kolmannessa luvussa käsitellään diplomityössä käytettävää tiedonhajautuksen teknologiaa ja teoriaa. Työssä käytettyä välikerrosarkkitehtuurin tuotetta analysoidaan suhteessa sen toteuttamaan standardiin ja johtamisjärjestelmän vaatimuksiin. Neljännessä luvussa kerrotaan työn käytännön osuudesta, jossa toteutettiin tiedonhajautus valitulla teknologialla ja tuotteella. Diplomityön tarkastelu ja mahdolliset

jatkokehityshankkeet on esitelty viidennessä luvussa. Diplomityön yhteenveto kerrotaan viimeisessä luvussa.

2. Toimintaympäristö

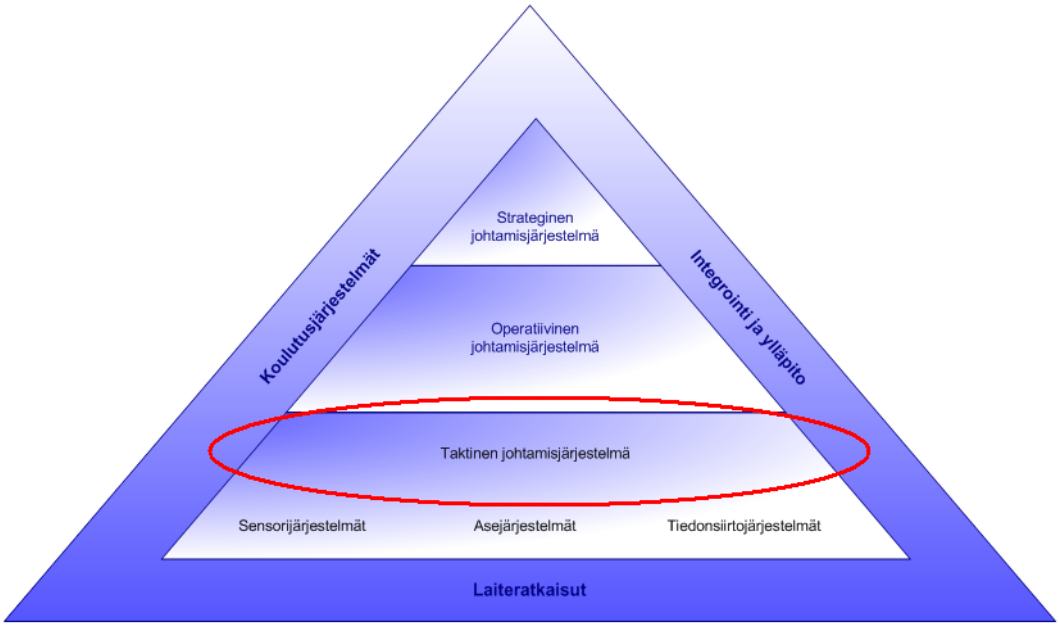
Tulevaisuuden verkostokeskeisen sodankäynnin sotatekniikat ovat työn kannalta suurin toimintaympäristöön vaikuttava tekijä. Toimintaympäristössä on keskeisenä osana ilmapuolustuksen johtamisjärjestelmä ja sen itsenäinen osa, toimipaikka. Luvussa käsitellään myös johtamisjärjestelmien tiedonhajautuksen haasteita, joihin tämän diplomityön tulisi antaa vastauksia.

2.1 Johtamisjärjestelmät

Suomen johtamisprosessit perustuvat puolustusdoktriiniin, jonka mukaisesti johtaminen on komentajakeskeistä ja tehtäväperustaista. Johtamisjärjestelmät ovat mukautuneet palvelemaan nykyisiä johtamisprosesseja. Jokaisen johtokeskuksen tulee olla tietoteknisesti itsenäinen kokonaisuus, mikä aiheuttaa tiivistä tiedonvaihtoa eri johtokeskuksissa olevien tietojärjestelmien välillä. Perinteisesti tietojärjestelmiä on lisäksi kehitetty jokaisen puolustushaaran omiin tarpeisiin, jolloin niiden lähtökohta verkostokeskeiseen sodankäyntiin ei ole ollut paras mahdollinen. Tämä on jo itsessään luomassa erityisiä tarpeita tiedon hajautukseen ja vaikeuttamassa tiedon hajautusta teknologiselta näkökannalta. (Puolustusvoimien Teknillinen Tutkimuslaitos, 2008)

Tämä työ liittyy hankkeeseen, jonka tavoitteena on nykyisen ilmapuolustuksen johtamisjärjestelmän uusiminen. Johtamisjärjestelmät jaetaan kolmeen tasaan: strateginen, operatiivinen ja taktinen johtamisjärjestelmä. Työssä käsiteltävä ilmapuolustuksen johtamisjärjestelmä on osa taktisen ja operatiivisen tasan johtamisjärjestelmää (Kuva 2). Johtamisjärjestelmien strateginen tasan tietojärjestelmät tuottavat strategisen tasan tilannekuvan sisältäen strategisen tiedustelun. Operatiivisen tasan tietojärjestelmät sisältävät operaatioiden

suunnittelemiseen ja johtamiseen sisältävät toiminnallisuudet. Taktisen tasan johtamisjärjestelmät ovat yleensä puolustushaarakohtaisia kuten tässä työssä käsiteltävä ilmapuolustuksen johtamisjärjestelmä. Taktisen tasan johtamisjärjestelmä sisältää myös vahvan liitynnän sensori- ja asejärjestelmiin. Ilmapuolustuksen johtamisjärjestelmä on käsitteenä lähempänä järjestelmien järjestelmää kuin yksittäistä paikallista järjestelmää.



Kuva 2 Johtamisjärjestelmien toimintaympäristö

Ilmapuolustuksen johtamisjärjestelmän tehtävänä on tuottaa Suomen ja sen raja-alueiden ilmatilannekuvaa ja tarjota johtokeskuksissa operoiville komentajille välineet johtaa ilmassa tapahtuvia operaatioita. Johtamisjärjestelmän yleiskuvauksessa tuodaan esille järjestelmään integroidut osapuolet (Kuva 3). Keskeisimmät toimijat ovat kiinteät ja liikkuvat johtokeskukset, sensorijärjestelmät, asejärjestelmät ja maasta käsin toimivat ilmatorjuntajoukot, joille johtamisjärjestelmä tarjoaa ilmatilannekuvaa.

Kuva 3 Yleiskuvaus johtamisjärjestelmän toimintaympäristöstä (Insta DefSec Oy, 2006)

Teknologisten haasteiden lisäksi johtamisjärjestelmiin kohdistuu myös ulkopuolisia uhkakuvia, jollaisina johtamisjärjestelmille voidaan nähdä yritykset lamauttaa järjestelmät epäsuoralla tulenkäytöllä tai elektronisen sodankäynnin menetelmillä. Edellisistä uhkakuvista johtuen johtamisjärjestelmien suojaaminen on välttämätöntä ja niiden tulee pystyä sekä toimimaan verkottuneessa ympäristössä että tarjoamaan taktisen tasan johtamisen palveluita itsenäisesti.

Verkottuneet johtamisjärjestelmät eivät ole olleet operatiivisessa käytössä vielä pitkään. Ensimmäisen sukupolven tietojärjestelmät ovat Puolustusvoimissa

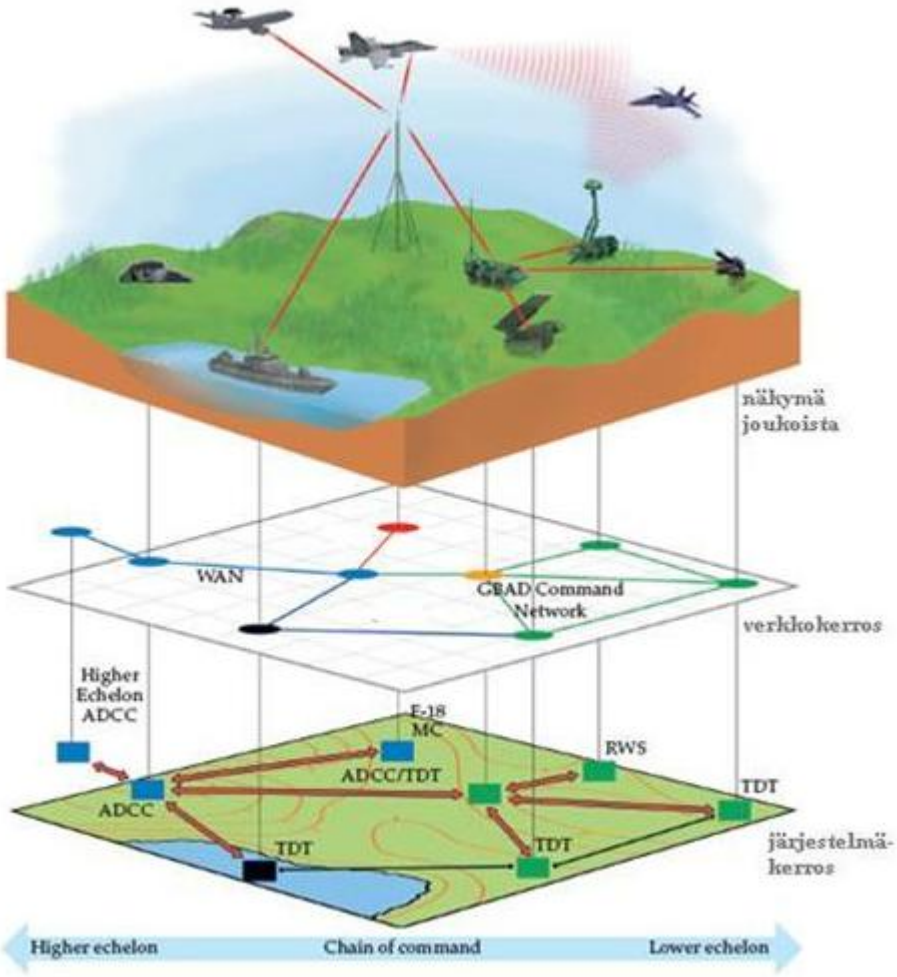
laajassa käytössä, mutta järjestelmien arkkitehtuurit ovat yleensä monoliittisia. Tämä on tilanne myös ilmavoimien johtamisjärjestelmässä. Johtamisjärjestelmän toiminnallisuus on kasvanut sen elinkaaren aikana merkittävästi ja järjestelmä on muuttunut ilmavoimien tarpeisiin valmistetusta taktisen tasan johtamisjärjestelmästä myös muiden ilmapuolustuksen aselajien käyttöön soveltuvaksi johtamisjärjestelmäksi. Siihen on toteutettu muiden aselajien tarpeita täydentäviä osakokonaisuuksia ja integroitu muiden aselajien järjestelmiä tarjoten kaikkien järjestelmien käyttöön yhteistä ilmatilannekuvaa. Yksi monista ilmavoimien johtamisjärjestelmään integroiduista järjestelmistä on ilmatorjunnan tulenkäytön johtamisen M90-järjestelmä (Jussila, 2004).

Nykyinen johtamisjärjestelmä on arviolta kehityskaarensa puolivälissä ja erittäin toimintakykyinen. Tulevaisuuden järjestelmän suunnittelu ja toteutus täytyy kuitenkin aloittaa riittävän ajoissa, jotta vaatimukset täyttävä järjestelmä olisi käyttöönoton yhteydessä valmis ja toimintakykyinen. Reaaliaikaisen taktisen tasan johtamisjärjestelmän tulee toimia luotettavasti ja täyttää kaikki viranomaisvaatimukset niin rauhan kuin kriisinkin aikana.

2.2 Verkostokeskeinen sodankäynti

Aikaisemmin sodankäynnin kriittisimmät osa-alueet olivat logistiikka ja tulenkäyttö. Tulevaisuudessa uudeksi kriittiseksi tekijäksi nousee verkostokeskeisen sodankäynnin hallitseminen. Verkostokeskeisessä sodankäynnissä järjestelmien tulee toimia erilaisissa verkkoympäristöissä, kuten oheisen kuvan (Kuva 4) verkkokerros esittää. Tiedonsiirtoverkot muodostuvat WAN, LAN ja radioverkoista, joiden avulla joukot kommunikoivat keskenään. Järjestelmäkerroksessa on kuvattu osa ilmatorjunnan johtamisjärjestelmistä (Air Defence Command & Control, ADCC) ja niiden liittymisistä muihin järjestelmiin. Liityntöjä on esimerkiksi liikkuvissa kohteissa oleviin päätteisiin (Tactical Data

Terminals, TDT), hävittäjissä oleviin järjestelmiin (F-18 MC) ja tutka-asemien järjestelmiin (Radar WorkStation, RWS).



Kuva 4 Verkostokeskeisen sodankäynnin kerroksellinen kuvaus (Insta DefSec Oy, 2006)

Hayesin mukaan sotilasoperaatioiden monimutkaisuus on lisääntynyt ja joukkojen yhteistoiminnan vaatimukset ovat kasvaneet. Verkottuneen taistelun avaintekijä on yhteentoimivuus, joka tarkoittaa järjestelmien ja joukkojen kykyä tuottaa ja käyttää toistensa tekemiä palveluja siten, että yhteistoiminta on tehokasta ja oikea-

aikaista (Hayes & Alberts, 2006). Yhteentoimivuus on vaatimuksena puolustushaarojen, operaatioihin osallistuvien eri maiden joukkojen ja eri turvallisuusalojen välillä. Yhteentoimivuus mahdollistaa tietoylivoiman syntymisen samalla, kun se yritetään kiistää vastustajalta. Tietoylivoimalla tarkoitetaan tässä yhteydessä kykyä kerätä, käsitellä ja välittää keskeytymätöntä tietovirtaa (Puolustusvoimien Teknillinen Tutkimuslaitos, 2008).

Johtamisjärjestelmien vaatimukset tulevat kasvamaan kapasiteetin ja dynaamisuuden osalta, tiedon ja tietolähteiden sekä yhteistyökumppanien määrän lisääntymisen takia. Tulevaisuuden johtamisjärjestelmät edellyttävät suurta tiedonsiirtokykyä ja automaattisesti mukautuvaa viestiverkkoa. Siirrettävien ja nopeasti liikkuvien joukkoyksikköjen ominaispiirteenä on liittyminen ja irtautuminen verkosta ja muista järjestelmistä (Jussila, 2004). Tämä luo erityisiä haasteita viestiverkon ja tiedon hajautuksen osalta. Pitää pystyä luomaan mekanismit, joilla taataan liikkuville joukoille tiedot käytettäväksi välittömästi verkkoon liittymisen jälkeen. Toisaalta myös muiden verkossa olevien järjestelmien tulee saada liikkuvien joukkojen tuottamat tiedot välittömästi käyttöön, kun joukot liittyvät yhteiseen verkkoon.

Yhteistoimintakumppaneiden kesken on olemassa rajoitettuja menetelmiä tiedon vaihtamiseen. Tarkkoja ja kaiken kattavia vaatimuksia tai standardeja tiedon hajauttamisen osalta ei kuitenkaan ole olemassa. Tässä diplomityössä ei keskitytä ratkaisemaan yhteistoimintakumppaneiden välistä tiedon hajauttamista. Vaikeutena on se, että vuonna 2009 valmistuneessa puolustuspoliittisessa selonteossa ei ole tarkemmin määritelty kuinka syvälle menevää integrointia ja yhteensopivuutta eri yhteistoimintakumppaneiden kanssa tullaan harjoittamaan. Kaikkea johtamisjärjestelmissä olevaa tietoa Suomen sisäisen turvallisuuden takaamiseksi ei luultavasti koskaan tulla jakamaan yhteistoimintakumppaneidenkaan kesken. Tulevaisuuden tietojärjestelmien kehityshankkeissa yhteistoimintakumppaneiden yhteensopivuus tulee kuitenkin

aiheuttamaan erityisiä tarpeita tiedon suodattamiseen tiedon hajautuksen yhtenä osana.

2.3 Toimipaikka

Toimipaikka on tämän diplomityön näkökulmasta keskeisin käsite. Se on fyysinen tai looginen kokonaisuus, joka tietojärjestelmien näkökulmasta pitää sisällään yhden itsenäisen järjestelmäkokonaisuuden. Yleisesti toimipaikka on joko Ilmavoimien pääjohto- tai apujohtokeskus. Liikkuvien joukkoyksikköjen tapauksessa kyseessä voi olla ilmatorjunnan tietojärjestelmä, joka on liittyneenä Puolustusvoimien runkoverkkoon. Toimipaikassa olevat tietojärjestelmät ja niiden palvelut voivat olla ajossa yhdessä tai useammassa tietokoneessa.

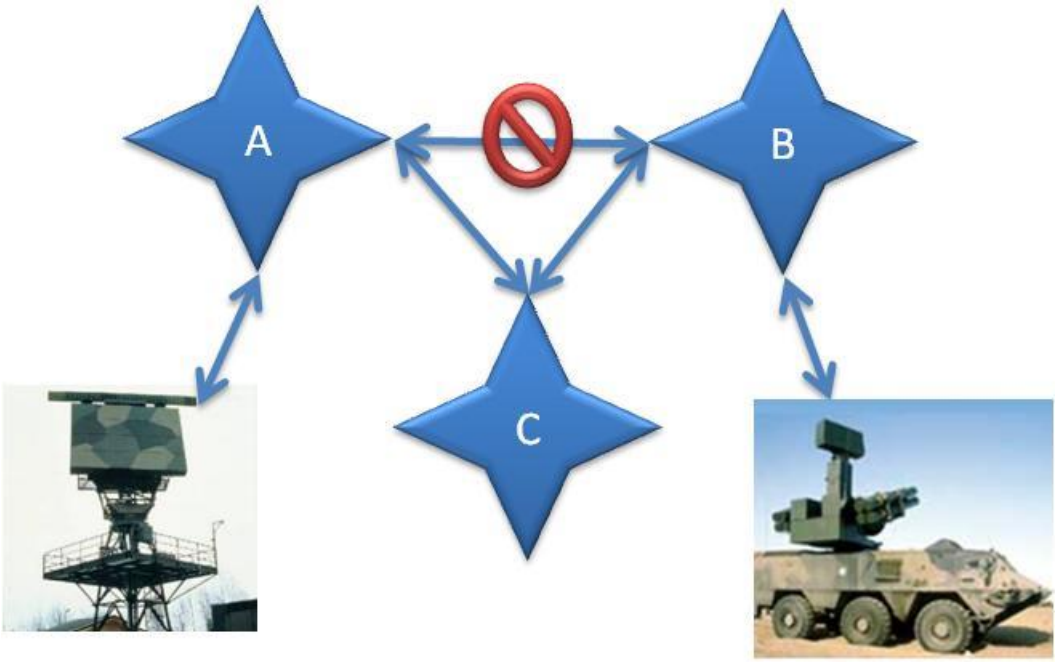
2.4 Reaaliaikaisuusvaatimukset

Reaaliaikainen tietojärjestelmä perustuu sekä tiedon oikeellisuuteen että ajalliseen täsmällisyyteen. Reaaliaikaisen sovelluksen ei siis tarvitse olla nopea vaan sen tulee vastata sille asetettuihin ajallisiin vaatimuksiin (Laplante, 2004). Reaaliaikainen tiedon hajauttaminen tarkoittaa tiedon siirtämistä yhdestä lähteestä yhteen tai useampaan kohteeseen ennalta määrätyssä ajassa. Tällä hetkellä on olemassa useita käytännön ratkaisuja ja tutkimuskohteita reaaliaikaiseen hajauttamiseen (Frolov et al., 2008).

Taktisen tasan johtamisjärjestelmällä johdetaan joukkoja hyvin lyhyen ajan suunnitelmilla, jolloin järjestelmän reaaliaikaisuusvaatimukset ovat suuria. Ilmatilannekuvan päivittämisessä puhutaan muutamista sekunneista, kun verrataan tietojärjestelmän esittämää tilannetta reaalimaailman tilanteeseen. Taktisen tasan johtamisjärjestelmää käyttäville henkilöille ei ole hyötyä nähdä

ilmatilannekuvaa minuutin viiveellä. Tällä tavalla esitetyssä ilmatilannekuvassa 1,8 machin nopeudella lentävä F/A-18 Hornet olisi edennyt 35 kilometrin päähän tietojärjestelmän osoittamasta paikasta. Tämä tarkoittaa, että järjestelmän näyttämän tiedon tulee olla mahdollisimman reaaliaikaista ja ehdottoman luotettavaa. Liian vanhaa tietoa ei saa esittää, ettei väärinymmärryksiä pääse syntymään.

Reaaliaikaisuusvaatimus aiheuttaa tietojärjestelmille erityisesti haasteita, kun johtamisjärjestelmässä oleva tieto joudutaan hajauttamaan usean toimipaikan yli. Tiedon hajauttamisen ongelmat eivät rauhanaikana ole yleisiä, mutta kriisitilanteessa voi syntyä tilanne, jolloin tietoverkon yhteyskatkon takia tieto joudutaan toimittamaan tiedon tarvitsijalle usean eri toimipaikan kautta (Kuva 5). Kuvan esimerkissä tiedon toimittaminen tutka-asemalta liikkuvaan toimipaikkaan joudutaan toimittamaan kolmen toimipaikan kautta. Tämä kuormittaa tietoverkkoa ja aiheuttaa tiedon hajautuksen kannalta viiveitä ja järjestelmään hetkellistä epäeheyttä ilmatilannekuvan osalta. Toimipaikka A näkee maalin ennen kuin se on ilmatorjuntajärjestelmän tiedossa, vaikka tiedon ensisijainen tarvitsija olisi ilmatorjuntajärjestelmä.



Kuva 5 Tiedon hajauttaminen usean toimipaikan yli

Diplomityön tulee etsiä ja pohtia tulevaisuutta varten myös vaihtoehtoisia tapoja toteuttaa tiedon hajautus toimipaikkojen välillä. Tarkoituksena olisi löytää menetelmät ja ratkaisut, joilla pystytään välittämään tietoa suoraan kaikille toimipaikoille tai joukkoyksiköille, jotka ovat tiedosta kiinnostuneita.

2.5 Oikeanlaisen tiedon hajauttaminen

Haasteena ei ole pelkästään reaaliaikaisuus vaan myös oikean tiedon välittäminen tiedon tarvitsijoille. Koska viestiverkkojen kapasiteetti on rajallinen, ei voida lähteä ajatuksesta, että kaikki tieto hajautetaan kaikille. Nykyisessä johtamisjärjestelmässä tiedon suodattaminen suotimien avulla toimii hyvin kahden toimipaikan kesken. Jos tietoa joudutaan hajauttamaan usean toimipaikan yli, pitäisi tiedon suodattaminen tapahtua kaikkien toimipaikkojen tarpeisiin liittyen. Tähän ei kuitenkaan ole olemassa toimivia mekanismeja vaan kaiken

tiedon tulee käytännössä hajautua toimipaikkojen välillä. Toisaalta nykyinen toimintatapa takaa varmimmin toimipaikkojen itsenäisen toimintakyvyn.

Palvelun laadun mekanismeina ovat nykyisessä johtamisjärjestelmässä tiedon priorisoinnin menetelmät. Johtamisjärjestelmä osaa priorisoida nopeasti muuttuvan tiedon, joka toimitetaan mahdollisimman reaaliaikaisesti. Hitaasti muuttuvien tietojen osalta johtamisjärjestelmä sallii tiedon hajauttamisessa enemmän viiveitä. Tulevaisuudessa on kuitenkin tarvetta pystyä priorisoimaan tietoa hienojakoisemmin. Myös muita tietoverkoissa käytettyjä palvelun laadun (Quality of Service, Qos) mekanismeja on tarkoitus ottaa käyttöön. Palvelun laadun mekanismeista on tarkempi kuvaus kolmannessa luvussa, jossa on esitelty diplomityössä käytetty reaaliaikaisen tiedon hajautuksen standardi.

2.6 Prosessoriarkkitehtuurin vaatimukset

Tietokoneiden prosessoriarkkitehtuurien muuttuminen viime vuosina on luonut haasteita nykyisten hyvin monoliittisten järjestelmien näkökulmasta. Uusissa tietokoneissa on useita rinnakkaisia moniydin-prosessoreita suorittamassa sovelluksia. Samalla suorittimien kellotaajuudet ovat alentuneet tai eivät ole merkittävästi nousseet niin kuin aikaisempina vuosina oli totuttu. Kellotaajuuden laskulla on haettu matalampia energiankulutuksia ja pienemmästä lämmöntuotosta johtuvaa vähäisempää jäähdyttämisen tarvetta. Vaikka tietokoneiden kokonaissuorituskyky on kasvanut, yksittäisen, yhdessä prosessissa ja säikeessä suoritettavan sovelluksen suorituskyky on todennäköisesti jopa hidastunut (Field, 2008). Tarkoitus oli löytää ratkaisu, joka hyödyntäisi paremmin rinnakkaisuutta. Tämä ei ole haaste pelkästään tiedon hajauttamiselle vaan tulevaisuuden tietojärjestelmien arkkitehtuurille yleisesti ottaen.

3. Data Distribution Service

Diplomityössä valittiin välikerrosarkkitehtuuriksi Data Distribution Service -standardi (DDS), jonka tarkoituksena on mahdollistaa reaaliaikainen tiedon hajauttaminen uudessa johtamisjärjestelmässä. Valinta perustuu Insta DefSec Oy:ssa tehtyihin tutkimuksiin eri teknologioiden sopivuudesta johtamisjärjestelmän välikerrosarkkitehtuuriksi. Yhtenä tutkimuksena on Peltolan diplomityö, jossa on analysoitu DDS:n toimivuutta hitaissa radioverkoissa (Peltola, 2008). Tässä luvussa esitellään myös valittu ja standardin toteuttava tuote, OpenSplice DDS.

3.1 Standardit

DDS on Object Management Groupin (OMG) kehittämä välikerrosarkkitehtuurin standardi, joka on suunniteltu hajautettujen reaaliaikajärjestelmien tarpeisiin. Järjestelmän keskeisenä ominaisuutena vastaaviin hajautusteknologioihin verrattuna ovat palvelun laatuun liittyvät ominaisuudet (QoS). Sotakelpoisen järjestelmän kehittämisen näkökulmasta DDS-teknologia tarjoaa mahdollisuuden toteuttaa vikasietoinen järjestelmä, jossa ei ole yhtä yksittäistä osaa (no single point of failure), joka voisi lamaannuttaa koko järjestelmän. Standardi tukee tietojärjestelmien liittymistä ja poistumista yhteisestä viestiverkosta. OMG on toteuttanut DDS:aan liittyen kaksi eri standardia:

1. Data Distribution Service for Real-time Systems (OMG, 2007)
2. The Real-time Publish-Subscribe Wire Protocol (OMG, 2008)

3.1.1 Data Distribution for Real-time Systems

Data Distribution Service for Real-time Systems on sovellustason ohjelmointirajapinnat määrittelevä standardi (OMG, 2007). Standardien tavoitteena on luoda valmistajariippumattomat rajapinnat DDS-teknologiaa hyödyntävien järjestelmien toteutukseen. Standardi mahdollistaa kaksi eritasoista ohjelmointirajapinnan toteutusta:

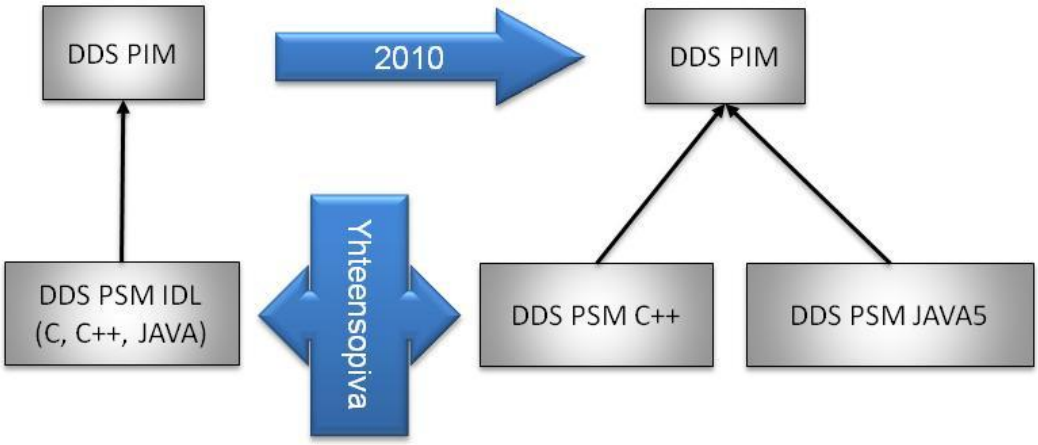
- Data-Centric Publish-Subscribe (DCPS), joka on standardin määritelmän mukaan pakollinen ja tarkoitettu alemman tason ohjelmointirajapintaa hyödyntäville sovelluksille.
- Data Local Reconstruction Layer (DLRL), joka ei ole standardin toteutuksessa pakollinen, mutta tarjoaa olio-ohjelmoinnin kannalta yksinkertaisemman korkeamman tason ohjelmointirajapinnan sovelluksille.

DCPS-rajapinnasta voidaan standardin mukaan toteuttaa minimitoteutus ja mahdollisia lisätoiminnallisuuksia, jotka on mainittu alla olevassa kuvassa (Kuva 6). DDS-standardi siis mahdollistaa eritasoisia DCPS-rajapinnan toteuttavia toteutuksia. Tämä antaa DDS-tuotteiden valmistajille mahdollisuuden toteuttaa uusia ominaisuuksia tuotteisiinsa vaiheittain. DDS-tuotteita hyödyntävien asiakkaiden näkökulmasta osittain standardin toteuttavat tuotteet vaikeuttavat evaluointia ja tuotteiden vertailua.

Kuva 6 DCPS profiilit (OpenSplice, 2009)

DCPS-rajapintakuvaus sisältää alustasta ja toteutuskielestä riippumattoman kuvauksen, Platform Independent Model (PIM), sekä Interface Definition Language (IDL) rajapintakuvauskieleen tehdyn OMG IDL Platform Specific Model (PSM) sovituksen. Käytännössä kaikki implementaatiot tukevat OMG IDL PSM:aa ja tuotteet toteuttavat esikääntäjät tukemilleen ohjelmointikielille. OMG:n mukaan DCPS-rajapinnan tavoitteena on tarjota tiedon hajauttamiseen tehokkaita, skaalautuvia, ennustettavuutta parantavia ja resursseja tehokkaasti hyödyntäviä mekanismeja (OMG, 2007).

DCPS standardin seuraava kehitysaskel tulee olemaan PSM-laajennukset, joissa tullaan tukemaan C++- ja Java5–ohjelmointikieliä (Kuva 7). Nämä laajennukset sisältävä standardin seuraava versio on tarkoitus julkaista vuoden 2010 aikana. Tavoitteena on, että uusi versio tulee olemaan yhteensopiva nykyisen julkaistun version kanssa (OMG, 2009).



Kuva 7 DCPS standardin seuraava kehitysaskel

DLRL-rajapinnan tavoitteena on tarjota näkymä DDS-tietomalleihin, kuin se olisi paikallinen tietomalli. DLRL toteuttaa relaatio–olio muunnoksen, jolloin on

mahdollista koostaa useammasta IDL-kuvauksen mukaisesta tietotyypistä sovelluksen tarpeisiin sopivat oliot. DLRL ei sulje pois DCPS-rajapinnan käyttöä, jos sen käyttöön suorituskyvyn takia olisi tarvetta. DLRL mahdollistaa olioille tietokannoista tutut päivitysoperaatiot (lue, luo, päivitä ja poista). DLRL tarjoaa myös muistinhallinnan olioille, jolloin oliot ovat paikallisessa välimuistissa sovelluksien käytettävissä.

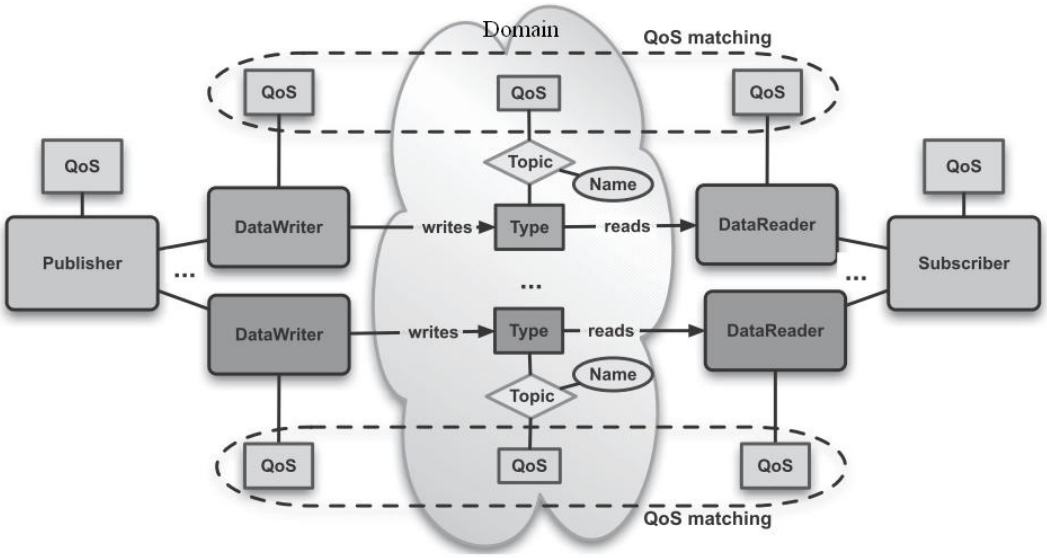
3.1.2 Wire-protokollastandardi

The Real-time Publish-Subscribe Wire Protocol on alemman tason protokolla, jolla DDS-toteutukset keskustelevat verkossa keskenään. Standardin tarkoituksena on taata yhteensopivuus eri toimittajien DDS-toteutuksien kesken. Standardi kuvaa toteutuskieleen riippumattoman kuvauksen, Platform Independent Model (PIM). PIM mahdollistaa protokollan yksikäsitteisen määrittelemisen ilman tietoliikenneprotokollien tuomia rajoitteita. PIM:n päälle on tehty sovitus PSM, jolla liitetään PIM mahdollisimman tehokkaasti User Datagram Protocol/Internet Protocol (UDP/IP) standardiin. Wire-protokollastandardi antaa toimijoille mahdollisuuden toteuttaa vain osajoukko standardista ja silti toteutukset voivat keskustella toisten DDS-toteutuksien kanssa yhteisessä verkossa. Tällöin tulee kuitenkin huomioida, että kaikki palveluiden laadulliset ominaisuudet eivät ole välttämättä käytettävissä.

Wire-protokollastandardi on osoittautunut olevan toimiva standardi, jonka monet toimittajat ovat omiin tuotteisiinsa toteuttaneet. Wire-protokollastandardin toteuttaneiden toimittajien yhteistoimintaa testaava demonstraatio toteutettiin maaliskuussa 2009. Testaus oli OMG:n järjestämä ja ensimmäinen laatuaan. Onnistuneeseen testaukseen osallistuivat PrismTech, RTI ja TwinOaks yhtiöt tuotteillaan (OMG, 2009).

3.2 Käsitteet

DDS-teknologiaan liittyy paljon käsitteitä (Kuva 8), joiden tunteminen on edellytyksenä ymmärtää diplomityön toteutus, sekä DDS-teknologian rajoitteet ja mahdollisuudet. Tässä luvussa esitellään keskeisimmät käsitteet, jotka perustuvat DDS-standardiin (OMG, 2007).



Kuva 8 DDS:n käsitemalli (Schmidth et al., 2008)

Domain (domain) on käsite, joka pitää sisällään yhden globaalin tietovaraston (Global Data Space, GDS). Kaikki järjestelmässä tarvittavat entiteetit liittyvät aina yhteen domainiin. Domain ei ole olio vaan konfiguraatio, joka yhdistää entiteetit. Järjestelmä, joka käyttää DDS-teknologiaa voi liittyä vain ja ainoastaan yhteen domainiin (Kuva 8).

Julkaisija (publisher) on entiteetti, jonka tehtävänä on tuottaa tietoa tietylle loogiselle alueelle. Julkaisija voi kuulua vain yhteen domainiin (Kuva 8) ja yhteen loogiseen alueeseen, mutta pystyy vaihtamaan loogista aluetta tarvittaessa. Yhdellä sovelluksella voi olla käytössä useampia tiedon julkaisijoita, mikäli sille

on erityisiä tarpeita. Käytännössä tämä tarkoittaisi erilaisia palvelun laatuun liittyviä vaatimuksia tai suuremman järjestelmän osa-alueiden riippuvuuksien poistamista. Yhdellä julkaisijalla voi olla useita tiedon kirjoittajia, joilla voidaan tuottaa näytteitä erilaisia aiheisiin.

Tilaaja (subscriber) on entiteetti, jonka tehtävänä on tilata sovellukselle tietueita tietystä loogisesta alueesta. Tilaaja voi kuulua vain yhteen domainiin (Kuva 8). Yhdellä sovelluksella voi olla käytössä useampia tilaajia aivan kuten julkaisijallakin. Yksi tilaaja voi omistaa useampia tiedon lukijoita, joilla voidaan lukea tietueita aiheista. Jotta tilaaja pystyisi tilaamaan tietyn julkaisijan tuottamia tietoja, julkaisijan ja tilaajan palvelun laadulliset asetukset tulee olla yhteensopivia ja molempien tulee kuulua samaan loogiseen alueeseen.

Aihe (Topic) koostuu tyypistä, yksilöivästä nimestä ja palvelun tasosta (Kuva 9).

Kuva 9 DDS Topic, mukailtu (Schmidth et al., 2008)

Aiheen tietotyypit on määritelty OMG:n IDL-kuvauskielessä. DDS-standardi mahdollistaa yhteen aiheeseen kirjoitettujen instanssien yksilöimisen avaintietojen perusteella, jotka kuvataan IDL-kuvauskielellä. Aiheelle pystyy määrittelemään laajat palvelun laadulliset ominaisuudet, jotka ovat tarkemmin esiteltyinä jäljempänä.

Kirjoittaja (Writer) on entiteetti, jonka omistaa julkaisija (Kuva 8). Kirjoittajan tehtävänä on tuottaa näytteitä vain ja ainoastaan yhteen määriteltyyn aiheeseen.

Kirjoittaja näyttelee keskeistä osaa, kun järjestelmän palvelun laadun suunnittelua tehdään. Käytännössä kirjoittajan palvelun laadullisten määrityksien perustaksi otetaan aiheen asetukset, joihin kyseiseen aiheeseen näytteitä tuottava kirjoittaja tekee sen tarvitsemat lisäykset.

Instanssikirjoittaja (Instance writer) on kirjoittajasta erikoistettu entiteetti. Instanssikirjoittajan tehtävänä on kirjoittaa näytteitä vain ja ainoastaan yhteen aiheeseen vain ja ainoastaan yhden instanssin muutoksista. PrismTechin mukaan instanssikirjoittaja on reaaliaikajärjestelmissä käyttökelpoinen entiteetti tarjoten suorituskykyisemmän tavan kirjoittaa globaaliin tietovarastoon nopeasti muuttuvien tietojen näytteitä (Prismtech, 2008).

Lukija (reader) on entiteetti, jonka omistaa tilaaja (Kuva 8). Lukijan tehtävänä on vastaanottaa näytteitä globaalin tietovaraston yhdestä aiheesta ja saattaa ne sovelluksen tietoon. Tiedon lukijan suunnittelussa tulee ottaa huomioon tietoa tuottavan kirjoittajan ja luettavan aiheen QoS-asetukset. Lukija ei voi vaatia laadultaan parempaa tietoa kuin mitä kirjoittaja pystyy aiheeseen tuottamaan. Lukijan QoS-asetuksia laadullisesti parempaa tietoa pystytään kuitenkin käyttämään hyväksi. Käytännössä lukija perii QoS-määritykset aiheelta lisäten omat vaatimuksensa määrityksiin.

Instanssilukija (instance reader) on lukijasta erikoistettu entiteetti. Instanssilukijan tehtävänä on lukea tietyn instanssin näytteitä yhdestä aiheesta. Muutoin instanssilukijan toiminnallisuus vastaa lukijan toimintoja.

Kuuntelija (listener) on entiteetti, jolla voidaan toteuttaa sovellukselle tarkkailija-suunnittelumallin mukainen toiminnallisuus, jossa sovellus saa välittömästi tiedoksi päivitetyt tiedot lukijalta. Kuuntelija-olioita voi olla rekisteröityneenä

yhdelle lukijalle useita. Kuuntelijan toteutus ei ole välttämätöntä, mutta reaaliaikaisuuden saavuttamiseksi käytännössä pakollinen ja suositeltava. Kuuntelijalle ei ole olemassa QoS-ominaisuuksia, koska sen toteutus tulee osaksi lukijaa.

Instanssi (instance) on aiheeseen julkaistava, uniikilla avaimella yksilöity tai yksilöimätön IDL-kuvauksella määritelty tietue. Samalle instanssille voi järjestelmässä olla useampia tuottajia. Mikäli käytetään useampaa kuin yhtä kirjoittajaa aiheen instanssia kohden, tulee niiden prioriteetit määrittää QoS-asetuksilla. Muutoin on vaarana instanssin tietojen osittainen häviäminen päällekirjoituksen yhteydessä. Suunnittelun kannalta on mielekästä, että yhdelle instanssille on yksi ensisijainen kirjoittaja. Mikäli aiheelle ei tyyppimäärittelyssä määritellä instansseja yksilöivää avainta, tuottaa DDS jokaiselle kirjoitettavalle tietueelle sisäisesti yksilöidyn avaimen. Instanssilla on olemassa elinkaari, jota voidaan määritellä kirjoittajan ja lukijan QoS-määrityksillä. Sovellukset saavat tiedoksi instanssissa tapahtuvat elinkaareen liittyvät muutokset ja pystyvät muutoksiin halutessaan reagoimaan.

Näyte (sample) on yhteen instanssiin kohdistuva päivitys. Kirjoittaja voi tuottaa yhteen aiheeseen tietylle instanssille sen avaintiedon perusteella näytteitä. DDS pystyy varastoimaan näytteitä instanssikohtaisesti QoS-määrityksien perusteella. Se antaa mahdollisuuden selvittää tietyn instanssin elinkaarta jälkikäteen. Kuuntelijoiden avulla tai näytteitä aiheesta hakemalla sovellukset reagoivat tapahtuneisiin muutoksiin. Näyte ei sisällä tietoa muutoksista edelliseen näytteeseen verrattuna. Mikäli sovelluksissa tulee tarvetta edellisten näytteiden tilan selvittämiseen, tulee käyttää QoS-määrityksiä tai sovelluksen tulee pitää kirjaa tapahtuneista muutoksista. Suunnittelun näkökulmasta parempi tapa on luoda riittävä määrä aiheita, jolloin uusi näyte sisältäisi pelkästään muutoksia eikä olemassa olevaa tietoa monistettaisi turhaan.

3.3 Palvelun laatu

Järjestelmäkehityksen keskeisimpänä ongelmana teknologiasta riippumatta on ollut palvelun laadun varmistaminen. Välikerrosarkkitehtuureilla ovat olleet haasteina se miten vastataan aikakriittisiin tarpeisiin, palvelun luotettavuuteen ja saatavuuteen (Corsaro et al., 2006). Nämä haasteet ovat erityisen merkittäviä tapahtumakriittisissä järjestelmissä kuten teollisuusprosessien hallinnassa tai puolustusteollisuuden järjestelmissä.

DDS on suunniteltu reaaliaikaisiin verkostokeskeisiin järjestelmiin, joissa oikea tieto toimitettuna liian myöhään muuttuu matkalla vääräksi ja järjestelmästä poistettavaksi tiedoksi. Päästäkseen tiedon oikeellisuuteen, järjestelmässä tulee huomioida verkon-, muistin- ja prosessoinnin tehokasta käyttöä (Schmidth et al., 2008). DDS-standardi tarjoaa 18 erilaista mekanismia palvelun laadun määrittämiseen ja menetelmillä on tarkoitus ohjata järjestelmän toimintaa. Tämä antaa hyvät mahdollisuudet suunnitella tehokas järjestelmä. Toisaalta monipuoliset palvelun laadulliset mekanismit tuovat tietojärjestelmiin monimutkaisuutta. Palvelun laadun määritteleminen ja hallinta luovat melkoisia haasteita suurissa järjestelmissä. DDS tarjoaa palvelun laadun mekanismeja alla olevan taulukon mukaisesti. Taulukossa on käytetty seuraavia lyhenteitä ja termejä:

- J, Julkaisija
- T, Tilaaja
- A, aihe
- K, Kirjoittaja
- L, Lukija.

Taulukko 1 Palvelun laadun politiikat, mukailtu (Schmidth et al., 2008)

QoS	Entiteetti	Yhteensopivuuden vaatimus	Ajonaikaisesti muutettavissa	Ryhmittely
DURABILITY	A, K, L	Kyllä	Ei	Tiedon pysyvyys
DURABILITY SERVICE	A, K	Ei	Ei	
LIFESPAN	A, K	-	Kyllä	
HISTORY	A, K, L	Ei	Ei	
PRESENTATION	J, T	Kyllä	Ei	Tiedon Välittäminen
RELIABILITY	A, K, L	Kyllä	Ei	
PARTITION	J, T	Ei	Kyllä	
DESTINATION ORDER	A, K, L	Kyllä	Ei	
OWNERSHIP	A, K, L	Kyllä	Ei	
OWNERSHIP STRENGTH	K	-	Kyllä	
DEADLINE	A, K, L	Kyllä	Kyllä	Tiedon oikea-aikaisuus
LATENCY BUDGET	A, K, L	Kyllä	Kyllä	
TRANSPORT PRIORITY	A, K	-	Kyllä	
TIME BASED FILTER	L	-	Kyllä	Resurssit
RESOURCE LIMITS	A, K, L	Ei	Ei	
USER_DATA	J, T, K, L	Ei	Kyllä	Konfigurointi
TOPIC_DATA	A	Ei	Kyllä	
GROUP_DATA	J, T	Ei	Kyllä	

Palvelun laadun mekanismien tarkemmat määritykset ovat OMG:n standardissa (OMG, 2007). Diplomityön kannalta keskeisimmät QoS-mekanismit ja niiden toimintaperiaatteet ovat seuraavat:

- *Durability service*, jolla määritellään tiedon pysyvyyteen liittyviä asioita. Osa tiedoista täytyy varastoida muistinvaraisesti (TRANSIENT), osa pysyvästi (PERSISTENT) ja osa voidaan käsittelyn jälkeen poistaa järjestelmästä automaattisesti (VOLATILE).

- *Lifespan,* jolla määritellään kirjoittalle sen kirjoittamien instanssien elinkaaren pituus. Tällä voidaan taata, ettei vanhentunutta tietoa käsitellä järjestelmässä.
- *History,* jolla määritetään kuinka monta näytettä instanssista säilytetään.
- *Reliability*, jolla voidaan määritellä tiedon hajautuksen luotettavuutta. Osalle tiedoista on tärkeää mennä aina perille. Joillakin tiedoilla elinkaari on niin lyhyt, etteivät ne tarvitse luotettavaa siirtotietä.
- *Partition,* jolla voidaan jakaa järjestelmän toiminnallisuuksia loogisiin alueisiin.
- *Destination order,* jolla määritellään näytteiden metatietona olevan aikaleiman hyödyntäminen eli käytetäänkö julkaisijan vai tilaajan aikaleimaa näytteiden järjestyksen määräävänä tekijänä.
- *Latency budget,* jolla voidaan määritellä maksimi viive instanssin kirjoittamisen ja lukemisen välille. Asetuksen arvo ei ole standardin näkökulmasta ehdoton noudatettava, mutta sillä voidaan ohjata järjestelmän sisäistä toimintaa.
- *Transport priority,* jolla pystytään latency budget QoS:n tavoin antamaan järjestelmälle pyyntö priorisoida tiedon välittämistä.
- *USER_DATA,* jonka avulla pystyy toteuttamaan tietuekohtaisesti esimerkiksi tietoturvaan liittyviä tarpeita.

3.4 OpenSplice DDS

OpenSplice DDS on Prismtech-yhtiön tuote, joka toteuttaa OMG:n DDS standardit. Tuotteesta julkaistiin keväällä 2009 avoimeen lähdekoodiin perustuva Community-versio. Community-versio on GNU Lesser General Public License version 3 (LGPLv3) lisenssin alainen tuote. Prismtechillä on Community-version lisäksi kaupalliset versiot (Kuva 10).

	Community	Compact	Professional	Enterprise
SOFTWARE				
Real-Time Pub/Sub (DCPS)	x	x	x	x
Interoperable Wire Protocol (DDSI)	x	x	x	x
Object Oriented Pub/Sub(DLRL)			x	x
Spike Absorber				x
Security				x
CONNECTORS				
SOAP-Connector			x	x
DBMS-Connector				x
TOOLS				
MDE PowerTools		x	x	x
Tuner		x	x	x
DDS TouchStone	x	x	x	x
Operating Systems				
Linux	x	x(1)	x(1)	x(1)
Windows	x	x(1)	x(1)	x(1)
Solaris	x	x(1)	x(1)	x(1)
AIX			x(1)	x(1)
VxWorks		x(1)	x(1)	x(1)
INTEGRITY		x(1)	x(1)	x(1)
Language Binding				
C	x	x	x	x
C++	x	x	x	x
Java	x	x	x	x
C#	x	x	x	x
LICENSE	**LGPL**	**Commercial**	**Commercial**	**Commercial**

Kuva 10 Prismtech OpenSplice DDS versiot (OpenSplice, 2009)

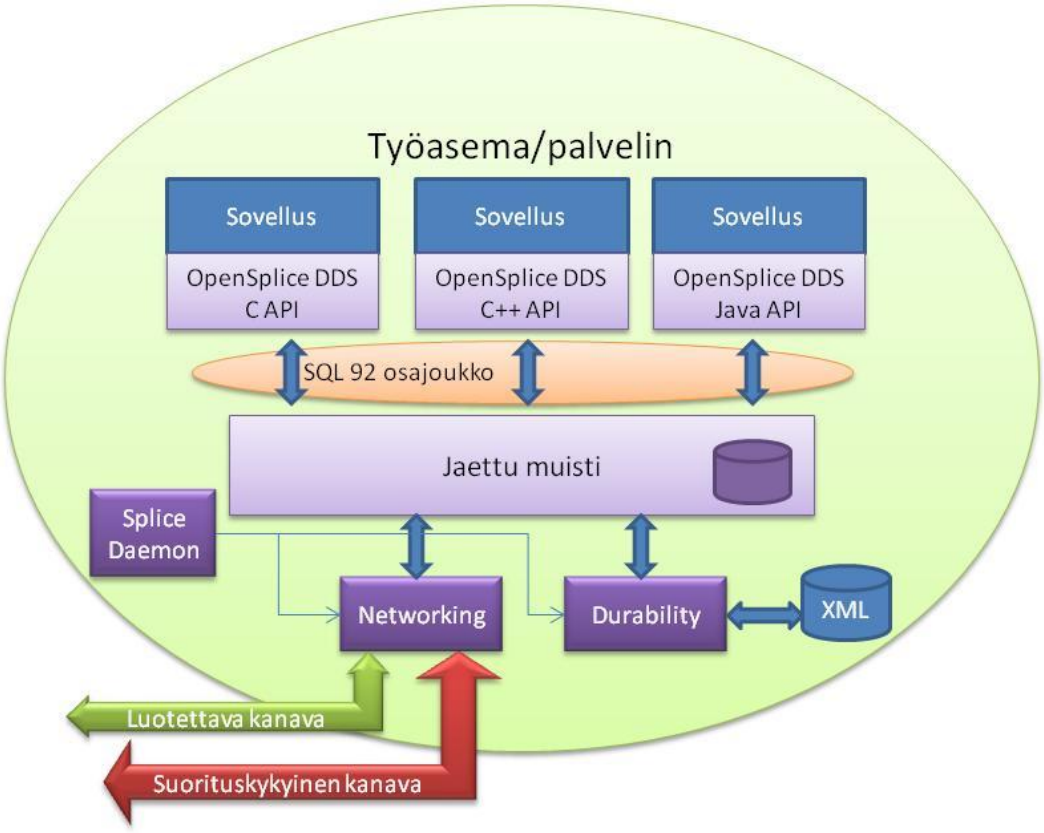
OpenSplice DDS perustuu Thales-yhtiön tekemään Splice-DDS-tuotteeseen, joka on vahvasti käytössä puolustusteollisuuden ratkaisuissa. Thalesin mukaan tuotetta on kehitetty yli kymmenen vuotta ja Thales on käyttänyt sitä sotalaivojen välikerrosratkaisuna (Thales, 2004). OMG:n DDS standardin toteuttamisessa Thales on ollut vahvasti mukana ja juuri siksi OpenSplice DDS on standardin kattavuudeltaan yksi markkinoiden parhaimmista tuotteista.

Diplomityössä käytettiin sekä Community-versiota että kaupallista Compact-versiota. Kaupallisten versioiden toteutus kattaa standardit laajemmin ja niiden mukana toimitetaan apuohjelmia, jotka helpottavat järjestelmien suunnittelua ja monitorointia (Kuva 10). Varsinaiseen tiedon välittämiseen työssä käytettiin Community-versiota. Diplomityön aikana työn alussa käytetty, keväällä 2009

julkaistu versio 4.1, päivitettiin uudempaan paremmin vaatimuksiin vastaavaan versioon 4.3.

3.4.1 Arkkitehtuuri

Työn kannalta merkittävimmät arkkitehtuuriin liittyvät komponentit ovat esiteltynä seuraavassa kuvassa (Kuva 11). OpenSplice DDS -tuotteen arkkitehtuuri sisältää muitakin tulevaisuuden kannalta kiinnostavia komponentteja, mutta ne ovat ainakin toistaiseksi saatavilla vain maksullisissa versioissa.



Kuva 11 Diplomityössä käytetty OpenSplice DDS -arkkitehtuuri

Arkkitehtuurissa keskeisin rooli on jaetulla muistilla ja sen tehokkaalla hyödyntämisellä. Arkkitehtuurissa on pyritty minimoimaan muistissa olevien tietueiden kopioiminen. Sovellukset voivat käyttää jaettua muistia suoraan, ilman että syntyy kopioita tietueista. Tämä vähentää sovelluksen muistin käyttöä ja parantaa suorituskykyä. Järjestelmä tarjoaa myös osittaisen toteutuksen Structured Query Language 92 –standardista (SQL). Tämä mahdollistaa tiedon hakemisen jaetusta muistista sisältöperusteisesti SQL-lausekkeiden avulla. Järjestelmän käynnistyessä luodaan yhteinen jaettu muisti kaikille DDS:n omille prosesseille ja säikeille, sekä DDS:aa hyödyntäville sovelluksille. Järjestelmän konfiguroinnista riippuen Splice Daemon –prosessi käynnistää tarvittavan määrän prosesseja palvelemaan sovelluksia. Keskeisimmät prosessit ovat Networking-prosessi ja Durability-prosessi. Networking-prosessin tehtävänä on tiedon välittäminen jaetusta muistista verkkoon ja verkosta jaettuun muistiin. Konfiguroinnista ja QoS-asetuksista riippuen käytetään erityyppisiä kanavia tiedon välittämiseksi verkkoon. Durability-prosessin tehtävänä on tiedon pysyvyyteen liittyvien palveluiden tarjoaminen. Pysyvyyden vaatima QoS-ominaisuus on Durability Service –QoS:lla, joka voidaan asettaa aiheelle, kirjoittajalle tai lukijalle (Taulukko 1). Tiedon pysyvyyteen liittyvä tallentaminen on joko muistinvaraista tai se voidaan tehdä tiedostoon, jolloin sen tallennusformaatti on eXtensible Markup Language (XML).

OpenSplice DDS pitää sisällään daemon-prosessin, jolle voidaan määrätä muiden prosessien valvontaan liittyviä tehtäviä. Jos jokin prosesseista jostain syystä kaatuu, on daemon-prosessin tehtävänä uuden palvelevan prosessin käynnistäminen. Tämä ominaisuus tuotteessa mahdollistaa, että erillistä valvontajärjestelmää tai palvelua ei tarvitse rakentaa.

3.4.2 Kattavuus

OMG:n DDS standardien näkökulmasta työssä käytössä oleva Community-versio toteuttaa DCPS-rajapinnan melko hyvin. Täydellisesti tuettuna ovat **Minimum**- ja **Ownership**-profiilit. Seuraavat profiilit ovat melkein täydellisesti tuettuna:

- **Content subscription** profiili, jonka toteutuksesta puuttuu haluttu ominaisuus, multitopic. Multitopic mahdollistaisi koosteiden tekemisen useammasta aiheesta käyttäen DDS standardin osittain tukemaa SQL-92 standardin mukaisia kyselyitä (OMG, 2007).
- **Persistence (durability)** profiili on toteutettu työn kannalta täydellisesti, mutta standardiin nähden siitä puuttuu TRANSIENT_LOCAL-toteutus. Se mahdollistaisi näytteiden tallentumisen vain näytteitä luovan järjestelmän paikalliseen muistiin. Työssä käytetään kuitenkin TRANSIENT-toteutusta, jossa tieto kopioituu kaikkien siitä kiinnostuneiden tilaajien paikalliseen muistiin ja on sieltä käytettävissä sovelluksen koko elinkaaren ajan.

DLRL-rajapinnan toteutusta ei ilmaisessa Community-versiossa ole saatavilla. DLRL-rajapinnan toteutuksen puutetta voidaan pitää merkittävänä. Se tulee aiheuttamaan hieman enemmän työtä ohjelmointivaiheessa. Työssä tehdään tarvittavat relaatio–objekti-muunnokset käsin.

Wire-protokolla on tuettuna Community-versiossa. Työssä ei kuitenkaan ole tarkoitus keskittyä tähän, koska työssä ja tuotteessa, johon työ liittyy, tullaan toistaiseksi käyttämään vain yhden toimittajan DDS-toteutusta. Konfiguraatiossa ei tulla toistaiseksi ottamaan Wire-protokollaa käyttöön, mutta sen olemassaolo on hyvä tiedostaa tulevaisuuden mahdollisia liityntöjä varten.

Kehitysympäristön vaatimukset ovat työn kannalta moninaiset. Tuotekehitys tehdään erilaisessa ympäristössä kuin tuotetestaus. Operatiivisella ympäristöllä on myös omat vaatimukset ajettaville sovelluksille. Vaatimuksista johtuen työssä käytettiin Community-versiota ja sen tukemia käyttöjärjestelmiä, jotka ovat:

- Linux
- Solaris 10
- Windows XP.

Community-version tukemat ja diplomityössä käytetyt ohjelmointikielet ja kääntäjät ovat:

- C/C++, kääntäjänä GNU Compiler Collection (GCC) versio 4.2.4
- Java5.

3.5 Muut hajautusteknologiat

DDS:n ohella tutkimuksissa on mietitty myös muita vaihtoehtoisia hajautusteknologioita. Parhaiten vaatimukset täyttäviä standardeja ovat Java Messaging Service (JMS) ja The Common Object Request Broker Architecture (CORBA). Tässä luvussa esitellään kevyesti edellä mainitut reaaliaikaisen tiedon hajauttamiseen soveltuvat teknologiat, koska niitä tullaan mahdollisesti käyttämään myös uuden johtamisjärjestelmän osana.

3.5.1 Java Message Service

Java Message Service (JMS) on Sun Microsystemsin kehittämä standardi Java-pohjaisiin järjestelmiin. JMS mahdollistaa sovellusten luoda, lähettää ja vastaanottaa sanomanvälitysjärjestelmän sanomia. JMS määrittelee standardin, jonka avulla sovellukset voivat käyttää eri valmistajien JMS-

viestinvälitysjärjestelmiä. JMS tukee point-to-point (PTP) ja julkaisija–tilaaja-mallin mukaista välitystapaa (Sun Microsystems, 2002).

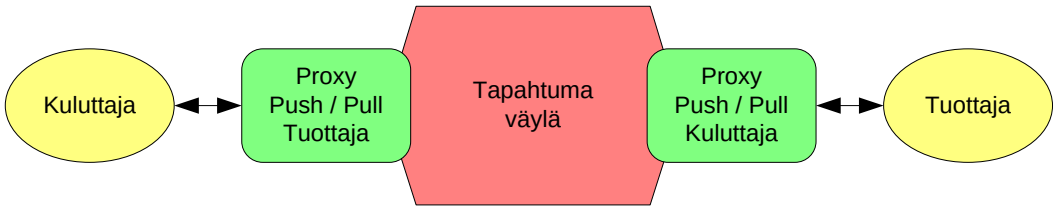
JMS käyttää yhtä tai useampaa palvelinta viestien säilömiseen ja lähettämiseen. Tämä mahdollistaa sanomien pysyvän säilömisen tapauksissa, joissa tilaaja sanoma ei saavuttanut tilaajaa tai välittämisessä tapahtui virhe tai uusi tilaaja ilmoittautui kiinnostuneeksi kyseisestä sanomasta. JMS välikerrosarkkitehtuuri perustuu best-effort palveluun, jossa järjestelmä yrittää parhaansa mukaan toimittaa sanomat perille ilman varsinaisia laatutakuita.

Olemassa olevat JMS toteutukset tukevat myös C++-ohjelmointirajapintaa, mikä mahdollistaisi nykyisen johtamisjärjestelmän liittämisen osaksi uutta johtamisjärjestelmää. Puutteena voidaan pitää, että JMS ei sisällä QoS-ominaisuuksia. JMS:n todellisena heikkoutena sodanajan johtamisjärjestelmäksi voidaan pitää tietoliikenteen keskittämistä JMS-palvelimeen, joka aiheuttaa Bondin (Bond et al., 2008) mukaan kriittisen kohdan järjestelmään (a single point of failure).

3.5.2 Real-Time CORBA

The Common Object Request Broker Architecture (CORBA) on OMG:n kehittämä välikerrosarkkitehtuurin standardi, joka mahdollistaa erilaisten olio-järjestelmien integroinnin. CORBA standardi määrittelee sovelluskehyksen, joka mahdollistaa sovellusten hajauttamisen riippumatta laitealustasta tai ohjelmointikielestä. CORBA:ssa osapuolten välinen rajapinta määritellään DDS:sta tutulla, mutta hieman eri tietotyyppejä tukevalla IDL-kielellä (OMG, 2004).

Reaaliaikavaatimuksiin OMG määrittelee Real-Time CORBA:n. Se tarjoaa kaksi eri lähestymistapaa toteuttaa kuluttaja–tuottaja palvelu, Push-mallin ja Pull-mallin (Kuva 12). Push-mallissa tuottaja alustaa yhteyden. Tuottaja lähettää dataa tapahtumaväylään, joka välittää datan kuluttajalle. Pull-mallissa kuluttaja alustaa yhteyden ja pyytää dataa tapahtumaväylältä. Tapahtumaväylä pyytää dataa tuottajalta. Tapahtumaväylän tehtävänä on vapauttaa kuluttaja ja tuottaja perinteisen CORBA:n synkronoiduista kutsuista (Frolov et al., 2008).



Kuva 12 Real-Time CORBA-arkkitehtuurin kuluttaja-tuottaja malli

Johtamisjärjestelmän suorituskyvyn kannalta Real-Time CORBA voisi olla vaihtoehto. Se ei kuitenkaan tarjoa kaikkia mahdollisesti tarvittavia QoS-mekanismeja. Real-Time CORBA:n tarjoama ohjelmointirajapinta ei myöskään tarjoa sellaista abstraktion tasoa, joka olisi tehokkaan ohjelmistotuotannon vaatimuksen mukainen ja joita kilpailevat teknologiat JMS ja DDS tarjoavat. Tulevaisuuden kannalta CORBA:n jatkokehittäminen lienee myös epävarmaa, koska CORBA:n standardoinut OMG on julkaissut uuden DDS-standardin hajautettuihin teknologioihin.

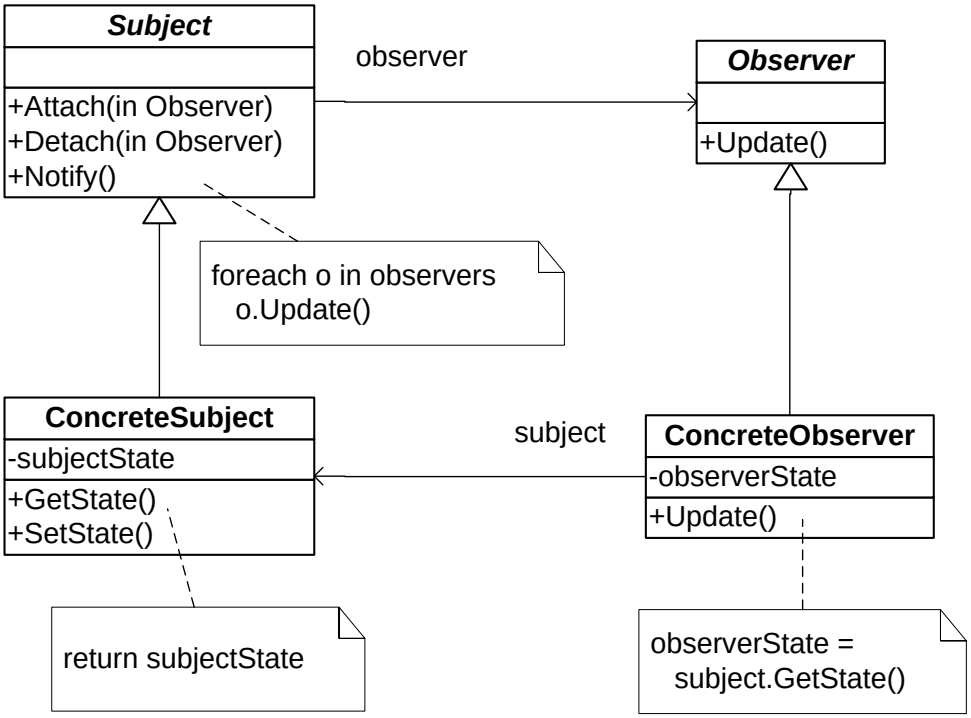
3.6 Reaaliaikajärjestelmien suunnittelumallit

Nykyisessä johtamisjärjestelmässä on hyödynnetty yleisesti tunnettuja suunnittelumalleja, joita myös tulevassa reaaliaikaisessa tiedon hajauttamisessa tullaan käyttämään. Reaaliaikaisen johtamisjärjestelmän tulee aktiivisesti välittää

tietoa sitä tarvitseville, eikä voida olettaa että tiedon tarvitsijan tarvitsisi kysyä tiedon olemassaoloa tai sen muutosta. Tällaisen vaatimuksen toteuttamiseen käytetään tarkkailijamallia (observer) (Gamma et al., 1995).

Tarkkailijamalli määrittää yhden suhteessa moneen riippuvuudet olioiden välillä siten, että olion tilan muuttuessa kaikki siitä kiinnostuneet oliot saavat muutoksesta automaattisesti tiedon. Mallin avainoliot ovat tarkkailija (observer) ja subjekti (subject). Subjektilla voi olla lukematon määrä tarkkailijoita. Tarkkailijat saavat tiedon heti, kun subjektin tila muuttuu. Tarkkailija voi täten saada subjektin tiedot ja synkronoida ne itseensä (Gamma et al., 1995).

Malli koostuu kahdesta abstraktista oliosta sekä niistä perityistä konkreettisista olioista (Kuva 13). Abstrakti subjekti-olio ottaa vastaan tilaukset tarkkailijalta ja tarjoaa rajapinnan tarkkailija-olion rekisteröitymiseen ja rekisteröinnin peruuttamiseen. Abstrakti tarkkailija-olio tarjoaa rajapinnan niille olioille, joille pitää ilmoittaa subjektissa tapahtuneista muutoksista. Konkreettisella subjekti-oliolla on tila, josta tarkkailijat ovat kiinnostuneita.



Kuva 13 Tarkkailijamallin rakenne (Gamma et al., 1995)

Subjekti-olio lähettää tarkkailijoille ilmoituksen aina, kun sen tila muuttuu ja tilan muutoksen johdosta voisi syntyä tilanne, että tarkkailijan ja subjektin tilojen välille syntyisi epäjohdonmukaisuutta. Saatuaan ilmoituksen tilan muutoksesta tarkkailija voi kysellä subjektilta tietoja. Tarkkailija käyttää tietoja päivittääkseen tilansa vastaamaan subjektin tilaa. Konkreettisella tarkkailija-oliolla on tieto seuraamastaan subjektista ja tila, jonka pitäisi pysyä synkronoituna subjektin tilan kanssa (Gamma et al., 1995). Reaaliaikajärjestelmässä tulee varmistua, että tarkkailijan käyttämä aika ei aiheuta liian pitkiä viiveitä, jotta subjekti pystyy ilmoittamaan muutoksestaan kaikille tarkkailijoille reaaliaikavaatimukset täyttävässä ajassa. Aikaa vievät tilan muutoksista johtuvat toimenpiteet tulee tarvittaessa suorittaa erillisissä prosesseissa tai säikeissä.

Tarkkailijamalli on käyttökelpoinen tiedon hajauttamisessa, kun välikerrosarkkitehtuurilta tulevia tapahtumia halutaan välittää sovelluskerrokselle.

Monet välikerrosarkkitehtuurit, DDS mukaan lukien, hyödyntävät tarkkailijamallin laajennusta, julkaisija–tilaaja-mallia (publish–subscribe). Käsitteet julkaisija–tilaaja-mallissa ovat nimetty eritavalla kuin tarkkailijamallissa. Julkaisija-tilaaja-mallissa subjektina toimii julkaisija ja tarkkailijaa kutsutaan tilaajaksi. JMS käyttää termeinä tuottajaa (producer) ja asiakasta (consumer).

Kaikki komponentit, jotka ovat riippuvaisia julkaisijan muutoksista, ovat sen tilaajia. Julkaisija ylläpitää rekisteriä aktiivisista tilaajistaan. Mikäli komponentti haluaa tilaajaksi, tulee sen rekisteröityä julkaisijalle. Tilauksen voi myös perua. Julkaisija–tilaaja-malli auttaa pitämään yhden suhde moneen olevat oliot synkronoituna, mutta tilaaja voi rekisteröityä usealle julkaisijalle. Julkaisija ilmoittaa tilansa muuttumisesta kaikille tilaajille (Buschmann et al., 1996).

4. Johtamisjärjestelmän tiedon hajautuksen toteutus

Diplomityön käytännön osuudessa suunniteltiin ja toteutettiin tiedon hajautus yhden toimipaikan sisällä. Tiedon hajautus toteutettiin nykyisen johtamisjärjestelmän ja uuden johtamisjärjestelmän välille yhden toimipaikan sisällä. Toteutuksen yhtenä osana suunniteltiin tiedon hajautus uuden johtamisjärjestelmän sisällä sekä suunniteltiin toimipaikkojen välisen tiedon hajauttamista.

4.1 Toimintaympäristö

Työssä on tarkoitus yhdistää OpenSplice DDS –tuotteella kaksi johtamisjärjestelmää. Tämä tarkoittaa, että joudutaan tukeutumaan molempien järjestelmien ympäristöihin. Nykyisessä johtamisjärjestelmässä työ toteutetaan C/C++-kielellä Solaris 10 käyttöjärjestelmälle. Uuden johtamisjärjestelmän toteutuskielenä on Java ja järjestelmä on lähtökohtaisesti ajoalustariippumaton. Valittuina ajoalustoina ovat Solaris 10 ja Linux.

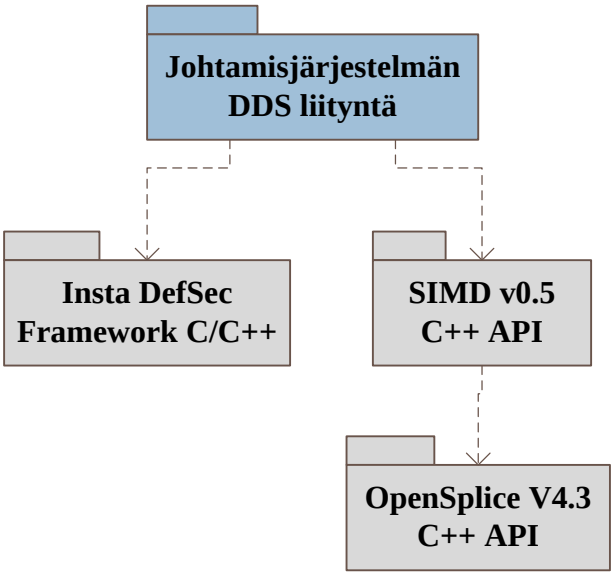
Työn toteutusvaiheen nopeuttamiseksi työssä käytetään SIMD-kirjastoa (SIMple DDS). SIMD on LGPLv3 lisenssin alainen C++:lla toteutettu template-luokkakirjasto. Kirjaston tarkoituksena on tuoda yksinkertaisempi ohjelmointirajapinta DCPS-kerroksen päälle ja mahdollistaa näin sovelluksen luominen yksinkertaisemmin ja nopeammin ja täten myös selkeämmin (Corsaro, 2009). SIMD ei estä DCPS-kerroksen käyttämistä. SIMD:n tarkoituksena on olla DDS-toimittajasta riippumaton, mutta toistaiseksi se tukee vain OpenSplice DDS -tuotetta.

Diplomityön käytännön osuuden toteutuksessa käytössä on SIMD versio 0.5. Tekniseltä toteutukseltaan SIMD perustuu template-luokkiin ja templaatteihin pohjautuvaan meta-ohjelmointiin. Versionumerosta voi kuitenkin jo ymmärtää, että kirjasto on vielä raakile, minkä takia virheiden korjaukseen kului hieman odotettua enemmän aikaa. Koska SIMD-kirjasto ei sisällä tilamaisia ominaisuuksia tai merkittäviä sisään rakennettuja toiminnallisuuksia, sitä voidaan pitää turvallisena käyttää. Sillä ei ole täten järjestelmän suorituskykyä heikentävää vaikutusta.

SIMD tuo eniten helpotusta DDS entiteettien hallintaan ja niille asetettavien QoS-asetuksien toteuttamiseen sekä poikkeusten hallintaan. SIMD käyttää vahvasti myös Boost-kirjastoja, joilla toteutetaan säikeiden luominen DDS-kuuntelijaa varten. Boost on jo ennestään Insta DefSecissä laajalti käytetty ja sen käyttäminen SIMD:n osana ei täten tuonut ylimääräistä työtä ympäristöjen asentamisen tai käytön opettelun osalta.

4.2 Ohjelmistoarkkitehtuuri

Nykyisen johtamisjärjestelmän liityntäohjelmisto toteutettiin järjestelmässä olevien vakiintuneiden rajapintamäärittelyiden mukaisesti. Liityntäohjelmiston perustoiminnallisuudet on toteutettu periyttämällä toteutusluokkia Insta DefSecin toteuttamasta ohjelmistokehyksestä (Kuva 14). Tietueiden julkaisemisessa globaaliin DDS tietovarastoon käytetään SIMD-luokkakirjaston template-luokkia.

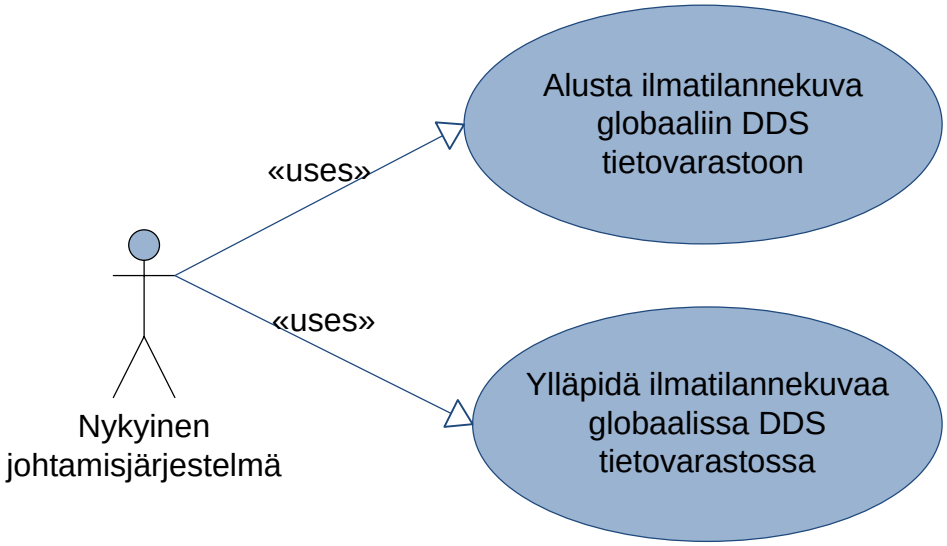


Kuva 14 Liityntäohjelmiston riippuvuudet moduuleista

Liityntäohjelmiston perustoiminnot sisältävät palvelun rekisteröinnin järjestelmän tiedonhallinnan ytimelle. Nykyisen järjestelmän taustaprosessien monitoroinnista vastaavalle sovellukselle rekisteröidään liityntäohjelmiston olemassaolo sekä ohjausvalikot liityntäohjelmiston toimintojen käyttämiseen.

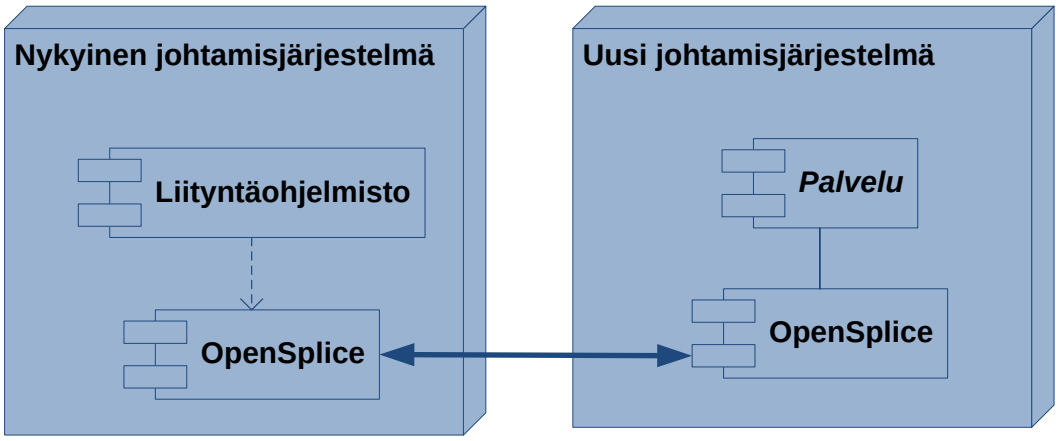
4.3 Toiminnallisuus

Liityntäohjelmiston tarkoituksena on tuottaa ilmatilannekuvaa nykyisestä johtamisjärjestelmästä uuteen johtamisjärjestelmään yhdessä toimipaikassa. Liityntäohjelmisto toteuttaa nykyisen johtamisjärjestelmän ilmatilannekuvassa olevien seurantojen alkulatauksen ja syntyneen ilmatilannekuvan ylläpitämisen uuteen johtamisjärjestelmään yhden toimipaikan sisällä (Kuva 15).



Kuva 15 Liityntäohjelmiston käyttötapaukset

Kun ilmatilannekuvaan vaikuttava seuranta päivitetään nykyisessä johtamisjärjestelmässä, se jaetaan heti DDS:n avulla tiedoksi uudelle johtamisjärjestelmälle (Kuva 16). Jokainen yksittäinen seurantaan vaikuttava muutos aiheuttaa uuden näytteen tuottamisen DDS tietovarastoon. Uuden ja vanhan järjestelmän välille ei rakenneta erillisiä tiedon synkronoinnin mekanismeja. Jos jostain syystä tiedon eheys järjestelmien välillä ei toteudu, suoritetaan liityntäpalvelun uudelleen käynnistäminen, jolloin alkulatauksen jälkeen ilmatilannekuva on yhteneväinen järjestelmien välillä.



Kuva 16 Liityntäohjelmiston toimintaperiaate

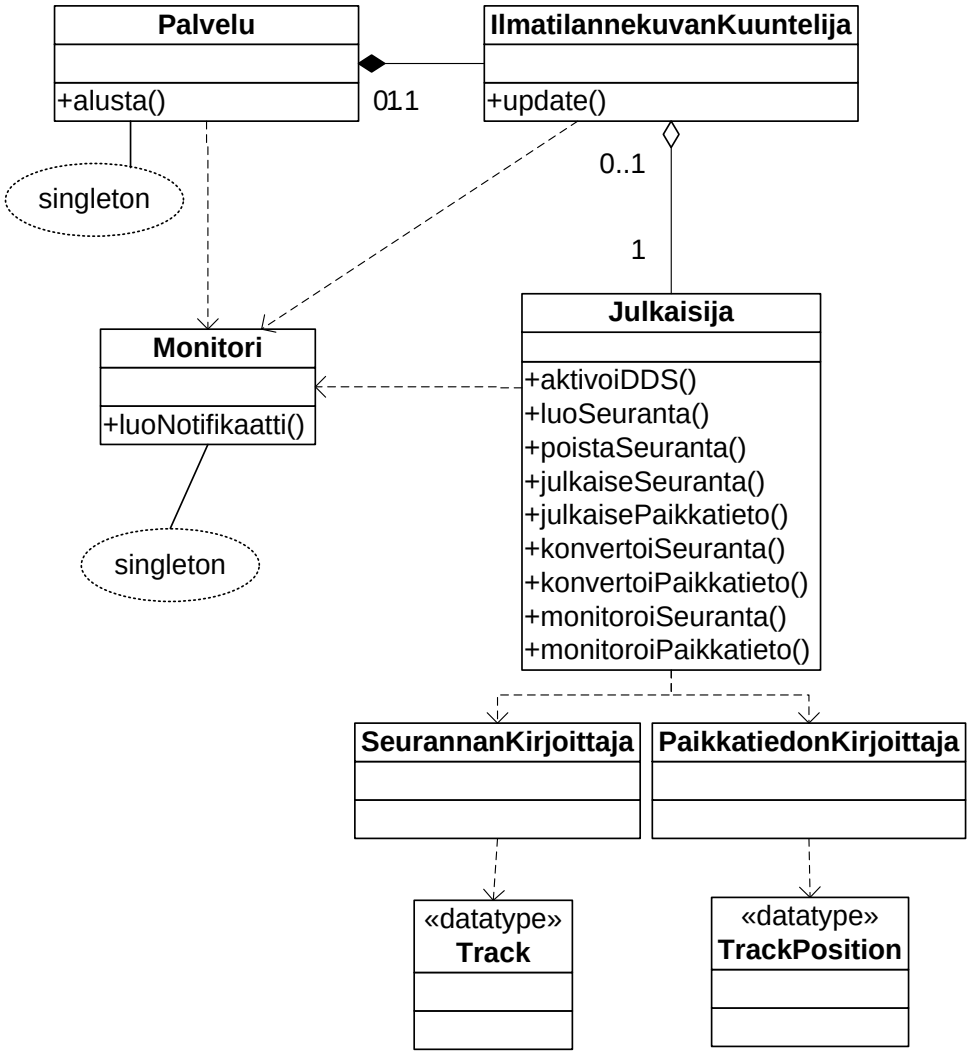
DDS standardina ei tue transaktionaalisuutta, mutta takaa tarvittaessa tiedon välittämisen. Uuden johtamisjärjestelmän vastuulle jää paikka- ja tunnistetiedon yhdistäminen. Standardi ei takaa välitettävien näytteiden järjestystä, kun näytteet liittyvät eri aiheisiin. Tilaajan tuleekin huomioida, että koostettavien olioiden tiedot voivat saapua satunnaisessa järjestyksessä. Mikäli käytössä olisi DLRL-rajapinta, ei yhdistämistä tarvitsisi toteuttaa sovelluksen logiikassa. Standardin mukaisen DCPS-rajapinnan tuki koostetuille aiheille (multitopic) mahdollistaisi tiedon sisältöön perustuvien, joukko-oppia hyödyntävien hakujen käytön. SQL-kielellä toteutettavan sisältöperustaisen tilaamisen käytön mahdollistaa myös nykyinen toteutus. Esimerkiksi tietyllä alueella olevien seurantojen hakeminen paikkatiedon perusteella onnistuu.

4.4 Tekninen toteutus

4.4.1 Liityntäohjelmiston luokkamalli

Liityntäohjelmiston toteutus perustuu Insta DefSecin valmiiseen ohjelmistokehykseen. Ohjelmistokehys sisältää ohjelmiston ytimen, tiedonhallinnan mekanismit ja ohjelmistojen monitorointiin liittyvät

toiminnallisuudet. Liityntäohjelmistossa toteutettu luokkamalli on karkeasti kuvattu seuraavassa kuvassa (*Kuva 17*):



Kuva 17 Liityntäohjelmiston luokkamalli

Palvelu-luokka toteuttaa liityntäohjelmiston rungon. Ohjelmistokehyksestä periytetty Monitori-luokka on singleton-suunnittelumallin mukaisesti toteutettu ja yhteisenä jaettuna resurssina liityntäohjelmiston luokille. Monitori-luokan tehtävänä on tuottaa nykyisen johtamisjärjestelmän monitoroinnille tilatietoja liityntäohjelmistosta ja DDS:n tietovarastoon julkaistavien tietojen sisältö. Ohjelmistokehyksen avulla liityntäohjelmisto liittyy nykyisen johtamisjärjestelmän tiedonhallinnan mekanismeihin tarkkailijamallin mukaisesti.

Se mahdollistaa IlmatilannekuvanKuuntelija-luokan rekisteröitymisen kuuntelemaan järjestelmässä tapahtuvia ilmatilannekuvan muutoksia ja käyttämään ilmatilannekuvaan liittyviä palveluita. IlmatilannekuvanKuuntelija-luokka omistaa Julkaisija-luokan, jonka avulla muuttuneiden seurantojen tiedot julkaistaan DDS:n globaaliin tietovarastoon. Julkaisemisessa käytetään kirjoittajia, joita on yksi jokaista tietovarastossa olevaa aihetta kohden.

4.4.2 DDS yhteyden käyttäminen

Julkaisija-luokka hyödyntää IDL-kuvauksen (LIITE 1 Viitteelinen IDL-kuvaus) tyypityksiä aktivoiDDS-metodissa, kun se luo Track ja TrackPosition aiheet globaaliin tietovarastoon ja asettaa aiheille QoS-määritykset. Aiheiden luomisen jälkeen luodaan tietojen kirjoittajat aiheille. Julkaisija-luokka tuottaa kahdella kirjoittajalla näytteitä ilmatilannekuvasta siten, että paikkatiedon muutokset kirjoitetaan TrackPosition-aiheeseen ja seurannan muut tiedot kirjoitetaan Track-aiheeseen.

Näytteiden kirjoittamisessa käytetään yleistä kirjoittajaa, joka ei ole instanssikohtainen. Tämä ei ole suorituskyvyn kannalta paras mahdollinen tapa, mutta tietomäärän ja käytettävissä olevan tiedonsiirtoverkon suorituskyvyn tuntien riittävä. Käyttämällä yleistä kirjoittajaa, jossa jokaisen näytteen yhteydessä kirjoittajalle toimitetaan instanssin avaintiedot mahdollistaa sen, että liityntäjärjestelmän ei tarvitse olla tietoinen välitettävistä seurannoista. Instanssikirjoittajaa käytettäessä tulisi olla kahvat kaikkiin instansseihin, joihin ollaan näytteitä kirjoitettu. Poikkeuksena edelliseen on instanssin poistaminen poistaSeuranta-metodilla. Instanssin poistaminen DDS:n *dispose*-toiminnolla edellyttää instanssikirjoittajan käyttämistä. Poistamista varten joudutaan luomaan väliaikainen instanssikirjoittaja.

4.4.3 Palvelun laatu

Järjestelmän ehdottomana vaatimuksena on tiedon reaaliaikaisuus. Vaatimuksena on myös tiedon hajauttaminen siten, että tiedon tulee olla käytettävissä myös esimerkiksi yhteyskatkojen aikana. Historiatietoa ei tarvitse tallentaa vaan siihen on tulossa myöhemmin erillinen toteutus. Liityntäohjelmisto käsittelee toistaiseksi vain nopeasti muuttuvaa tietoa. Tiedon oikeanaikaisuuteen ei tarvitse vielä keskittyä, koska siitä pitää huolen toistaiseksi nykyinen johtamisjärjestelmä.

DDS-teknologian valinnassa oli suuri painoarvo palvelun laadulla. Liityntäohjelmistossa tehtiin seuraavat standardin oletusarvoista poikkeavat QoS-asetukset:

- **Durability service** asetetaan molemmille aiheille arvoon TRANSIENT. Se takaa, että samat näytteet ovat saatavilla molemmissa järjestelmissä riippumatta toisen järjestelmän tilasta. Tämä pitää huomioida myös uuden järjestelmän lukijoissa, jotta järjestelmän käynnistyessä kaikkien seurantojen tiedot ovat saatavilla.
- **Lifespan** asetuksella määritetään seurantojen elinkaaren pituus. Tämä on hyvä erityisesti yhteyskatkojen tai järjestelmien ylikuormituksen aikana. Tällä pystytään estämään, ettei järjestelmässä prosessoida enää vanhentunutta tietoa. Erityisesti seurannan paikkatieto on tämänlainen tieto.
- **History** asetetaan arvoon 1, jolloin vain viimeinen näyte jätetään jaettuun muistiin, ja annetaan tiedoksi sovelluksille. Tämä voi aiheuttaa sen, että jos yksi instanssi saa useita näytteitä samanaikaisesti, ei kaikista tule tietoa tilaajalle. Vain uusin näyte annetaan tiedoksi tilaajalle.
- **Reliability** asetetaan kriittisille aiheille. Tässä työssä oleva Track-aihe pitää sisällään esimerkiksi tunnistustiedon. Järjestelmissä ei saa päästä tapahtumaan tilannetta, jossa viholliseksi tunnistettu lentokone pääsisi etenemään Suomen ilmatilassa tunnistamattomana kohteena.

- **Partition** asetetaan julkaisijalle ja tilaajalle. Tällä ominaisuudella eristetään vanhan ja uuden järjestelmän aiheet muusta globaalista tietovarastosta ja luodaan liityntää varten oma looginen tietovarasto.
- **Destination order** asetettiin käyttämään lähdejärjestelmän aikaa. Tämä edellyttää, että jokaiselle näytteelle minkä julkaisija kirjoittaa, täytyy asettaa aikaleima metatiedoksi. Suunnittelun kannalta on järkevää, että tiedon omistava osapuoli määrittää näytteiden järjestyksen tilaajalle. Tämä takaa myös kronologisen järjestyksen näytteille verrattuna vaihtoehtoon, jossa järjestys määräytyisi vasta tilaajan prosesseissa.

4.4.4 Konfigurointi

Valmiin tuotteen ottaminen osaksi johtamisjärjestelmän tiedon hajautusta on toteutusta nopeuttava ja laatua parantava valinta. Vaikka ohjelmiston suunnittelussa, toteutuksessa ja testauksessa säästetään aikaa, tulee ylimääräisenä työnä tuotteen hallinta ja konfiguroiminen. DDS-standardi itsessään ei ota kantaa miten standardi tulee toteuttaa, ainoastaan millaisia rajapintoja ja toimintoja sen tulee sisältää. Tämä tarkoittaa, että jokainen DDS-toteutus on omanlaisensa ja eri tavalla konfiguroitava.

OpenSplice DDS –tuotteen konfigurointi määritellään XML-formaatilla. Konfiguraatio ilmaistaan käyttämällä Uniform Resource Identifier (URI) muotoa. Tuotteen mukana tulee ohjelmisto (osplconf), jolla järjestelmän konfigurointia voidaan toteuttaa, ellei halua käyttää perinteistä XML-editoria. Tuotteen mukana tuleva ohjelmisto antaa kuitenkin selkeämmän näkymän konfigurointiin. Se ohjeistaa konfiguroinnissa ja estää karkeat virheet (PrismTech, 2009).

Liityntäjärjestelmän konfiguroinnissa keskeisimmät konfiguroitavat asiat liittyvät Networking-prosessiin (Kuva 11), jonka tehtävänä on toteuttaa tiedonsiirto

muihin järjestelmän tietokoneisiin. Liityntäohjelmisto käyttää multicast-osoitetta yhteyksien luomiseen (discovery) ja tiedon välittämiseen. Durability-prosessille ei järjestelmässä tarvitse konfiguroida talletustiedostoa, koska muistinvarainen tiedon pysyvyys (TRANSIENT) on riittävä. Järjestelmälle muita olennaisia konfiguroitavia asioita ovat jaetun muistin riittävä koko ja virheistä toipuminen.

OpenSplice DDS -tuotteen ja sitä hyödyntävien järjestelmien tulee käyttää yhteneväisiä URI-tiedostoja. URI-tiedosto tarvitaan liityntäohjelmiston käynnistämisessä ja DDS-domainin luonnissa. OpenSplice DDS parsii URI-tiedostoista tarvittavat tiedot ja alustaa järjestelmän niiden mukaisiksi.

4.5 Toimipaikkojen välisen tiedon hajautuksen vaihtoehdot

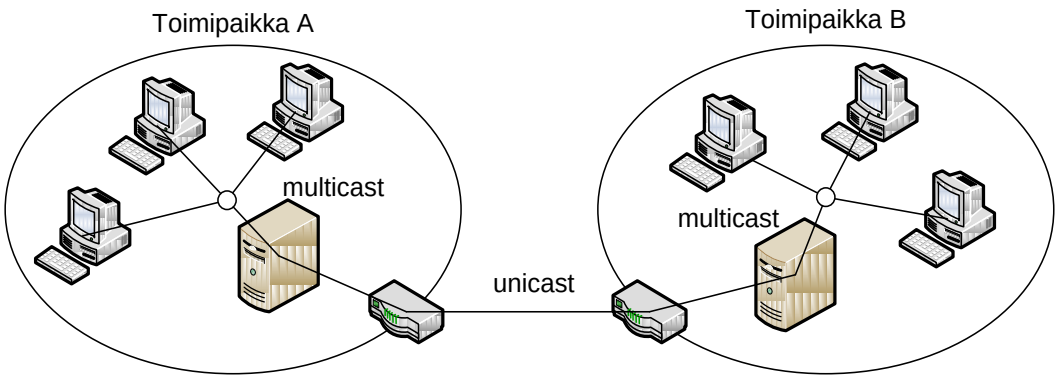
DDS-teknologian hyödyntäminen tulevaisuudessa toimipaikkojen välisen tiedon hajauttamiseen tarjoaa useampia toteutusmahdollisuuksia:

- Rautatason toteutus, jossa reitittimien asetuksia muuttamalla saadaan multicast-osoitteisiin toimitetut viestit siirtymään aliverkosta toiseen.
- Unicast-toiminnon käyttäminen, jossa toimipaikkojen väliset yhteydet määriteltäisiin kiinteästi DDS-konfiguraatiolla.
- Virtual local area network (VLAN) teknologian hyödyntäminen.

Reitittimien konfiguroimisen osalta ollaan oltu hieman varovaisia. Konfigurointi olisi suuritöinen ja käytännössä mahdoton toteuttaa. Konfiguroinnin seurauksena syntyisi kaikkien toimijoiden kesken yksi suuri yhteinen globaali tietovarasto. Hajautuksen näkökulmasta tämä olisi hieno vaihtoehto. Wide Area Network (WAN) kuormittuisi kuitenkin huomattavasti tietoliikenteestä ja useiden reitittimien yli kulkevan tiedon viive järjestelmien välillä voisi kasvaa liian

suureksi vaatimuksiin nähden. Tämä on kuitenkin yksi mahdollisuus, joka tulevaisuuden tutkimushankkeissa tulee selvitettäväksi.

Unicast-tiedonsiirron hyödyntäminen, mikä tuli osaksi OpenSplice DDS -tuotetta viimeisimmän version V4.3 mukana, on jo nyt varteenotettava vaihtoehto tiedon hajauttamiseen toimipaikkojen kesken. Toteutuksen näkökulmasta tämä tarkoittaisi julkaisija–tilaaja parin muodostamista toimipaikkojen välille. Toimipaikkojen välisten prosessien tehtävänä olisi halutun tiedon tuottaminen toimipaikan lähiverkkoon multicastilla (Kuva 18). Unicast-tiedonsiirtoa hyödyntävän vaihtoehdon haittapuolena olisikin toteutukseen tarvittava suuri työmäärä.



Kuva 18 Toimipaikkojen välinen tiedonhajautus unicast-tiedonsiirrolla

Unicast-toiminnallisuuden käyttäminen tuottaa monimutkaisemman ja ennalta toimipaikkojen tiedot määrittävän URI-konfiguraation (LIITE 2 URI-konfiguraatio unicast-tiedonsiirrolle). Siinä tulee ottaa multicast-tiedonsiirtoon verrattuna huomioon IP-pakettien tarpeeksi suuri elinkaari Time-To-Live (TTL), koska tiedossa ei ole varmuudella pakettien välittämiseen tarvittavien reitittimien lukumäärää.

VLAN-teknologian hyödyntäminen on käsitteenä melko lähellä rautatason konfigurointia, mutta se toteutettaisiin ohjelmistoilla. Teknologia perustuu ohjelmistoihin, jotka kykenevät toteuttamaan paikallisen lähiverkon (Local Area Network, LAN) WAN:n päälle. VLAN:lla on samat ominaisuudet, kuin fyysisesti toteutetulla lähiverkolla (IEEE Computer Society, 2006). VLANin ominaisuudet mahdollistavat DDS-teknologian hyödyntämisen tiedon hajauttamiseen käyttämällä broadcast- ja multicast-osoitteita.

5. Johtopäätökset

Diplomityössä perehdyttiin DDS-standardiin ja sen sopivuuteen tiedon hajauttamiseksi johtamisjärjestelmässä. Käytännön osuudessa suunniteltiin ja toteutettiin liityntäohjelmisto nykyisen- ja uuden johtamisjärjestelmän välille. Tässä luvussa käsitellään diplomityön vahvuuksia ja heikkouksia ja pohditaan millaisia jatkokehityshankkeita olisi järkevää toteuttaa.

5.1 Tarkastelua

Työn alkuperäisenä ajatuksena oli tutkia DDS-teknologian käyttöä toimipaikan sisäisen ja toimipaikkojen välisten tietojen hajauttamiseen. Hankkeiden aikatauluista johtuen toimipaikkojen välinen tiedon hajautus tullaan vielä toistaiseksi hoitamaan vanhalla menettelyllä, joten se osuus jäi diplomityössä teorian asteelle. Toimipaikan sisäisessä tiedonhajauttamisessa toteutettiin liityntäohjelmisto vanhan ja uuden johtamisjärjestelmän välille. Liityntäohjelmiston avulla tuotetaan ilmatilannekuvaa uuteen johtamisjärjestelmään.

OpenSplice DDS on tuotteena jo ollut kaupan operatiivisissa järjestelmissä, mutta DDS-standardina vielä kuitenkin melko uusi. Tästä johtuen standardista on vain vähän luotettavaa kirjallisuutta tai tieteellisiä artikkeleita saatavana. Rajapintadokumentaatioita löytyy, mutta saatavilla ei ole kirjallisuutta, josta kävisi ilmi, miten järjestelmiä tulisi DDS-teknologialla suunnitella ja toteuttaa. Vaikka työ on toteutettu ja teknologia on jo kuluneen vuoden aikana tullut tutuksi, on moni asia edelleen epäselvää. Valitettavan monen ongelman joutuu oppimaan kantapään kautta.

Liityntäohjelmiston suunnittelun suurimpana haasteena oli vastaavien DDS-teknologioita hyödyntävien esimerkkiohjelmistojen puute. Niitä ei ollut. Toteutus itsessään onnistui tavoiteaikataulussa. Se oli osaksi SIMD-kirjaston ansiota. DCPS-rajapinnan toteuttama C++ API on monimutkainen eikä se ole puhdas oliototeutus. Pelkästään sitä käyttämällä toteutuksesta olisi tullut monimutkainen ja toteutusaikataulusta haasteellinen. Valitettavasti Java:lle ei ole vielä olemassa vastaavaa toteutusta, kuin mitä SIMD on C++:lle, mutta tällainen on mitä ilmeisimmin vuoden 2010 aikana syntymässä.

Työn alussa keväällä 2009 käyttöön saatiin ensimmäinen ilmainen OpenSplice DDS Community-versio 4.1. Työn edetessä siirryttiin käyttämään versiota 4.3, jossa oli tuki myös vaadittavalle Solaris 10 käyttöjärjestelmälle. Kääntäjänä oli tuettu vain Sun Studio 10 eikä the GCC 4.2.4, joka oli vaatimuksena työlle. Insta toteutti tarvittavat muutokset ja julkaisi GCC 4.2.4 -tuen yhteisön käyttöön.

OpenSplice DDS ympäristöjen konfiguroiminen on myös haasteellista etenkin, kun sen monitoroiminen ei ole yksinkertaista ja helppoa. Tuotantoympäristön yksinkertaisuus ja vaatimukset kuitenkin helpottivat tilannetta. Ongelmia olikin kehitysympäristössä, jossa verkkotopologian monimuotoisuus aiheutti ongelmia. Osana ongelmien ratkaisua oli apuna Internetissä toimiva OpenSplice DDS -yhteisö.

5.2 Jatkokehityksen aiheita

OpenSplice DDS on open source tuote ja Insta DefSec:lla on ollut ajatuksia sitoutua sen kehitykseen yhteisön jäsenenä. Se vaatii kohdennettua resursointia hankkeisiin, mutta onnistuessaan tuo lisäarvoa paremman tuotteen, mutta ennen kaikkea paremman osaamisen muodossa.

Ehdoton tarve tulevaisuuden projekteja ajatellen on selkeä **Java API**. OMG:n uusi DDS-standardi tulee ilmeisesti toteuttamaan Java5 PSM:n DCPS-rajapinnasta. Se voisi olla hyvä lähtökohta toteutukselle. Toinen vaihtoehto on alkaa kehittämään kerrosta nykyisen Java API:n päälle ja tähän suuntaan ollaan pakon sanelemana toistaiseksi menossa.

Multitopic tuen luominen Content subscription profiiliin olisi tarpeellinen hanke ja luultavasti tulee toteutetuksi yhteisössä. Ajankohdasta vain ei ole tietoa. Periaatteessa tämän toiminnallisuuden saa aikaiseksi toteuttamalla koostavan aiheen. Koostavassa aiheessa sovellus kokoaa tietoa eri aiheista käyttäen DDS-standardin tukemia SQL-hakuja ja luo tuloksista halutun kokonaisuuden. Sovellustasolla tämä aiheuttaa melko monimutkaisen toteutuksen, jonka jättäisi mielellään tuotteen vastuulle, kun standardikin sen sinne osoittaa.

Transaktioiden tuki tarvitaan jossain vaiheessa, kun tietomallit monimutkistuvat. Transaktion toteutus käytännössä olisi erillisen aiheen luominen, joka tietäisi transaktioon kuuluvien näytteiden aiheet ja osaisi kuuntelijassa koota kaikki yhteen transaktioon kuuluvat näytteet. DLRL-kerros toteuttaa tämän kaltaisen toiminnallisuuden säilömällä välimuistiin aiheiden näytteet ja luomalla niistä olioita. DDS standardi ei sisällä transaktioita eikä niitä varmasti DDS:n käyttötarkoitus tietäen ole tulossakaan. Sovelluskehityksessä ne vain ovat usein

välttämättömiä. Ilman transaktioita tietomallien suunnittelu on huomattavasti haastavampaa ja tuottaa todennäköisesti monimutkaisempia ratkaisuja.

DLRL-rajapinnan tulevaisuus on mielenkiintoinen. Se on olemassa OpenSplice DDS -tuotteessa, mutta ei Community-versiossa. Aikaisemminkin on nähty, että kaupallisesta versiosta on tuotu ominaisuuksia ilmaiseen versioon ja toivoa sopii, että tämän kanssa käy vastaavasti. DLRL lisäisi huomattavasti tuottavuutta ja selkeyttäisi lähdekoodia. Suorituskykymielessä sen käytössä tulee kuitenkin olla varovainen.

6. Yhteenveto

Diplomityössä tavoitteena oli tutkia, suunnitella ja toteuttaa johtamisjärjestelmän tiedon hajautus DDS-teknologialla. Käytännön toteuttamisen osaksi tuli nykyiseen johtamisjärjestelmään toteutettava liityntäsovellus. Tähän liittyvät vaatimukset tulivat suuremmasta hankekokonaisuudesta, jonka päämääränä on toteuttaa uusi ilmapuolustuksen johtamisjärjestelmä. Keskeisimpänä vaatimuksena oli reaaliaikaisen ilmatilannekuvan välittäminen nykyisestä johtamisjärjestelmästä uuteen johtamisjärjestelmään. Toteutuksen teknologiaksi valittiin OMG:n standardoima DDS. Se on suunniteltu reaaliaikaisen tiedon hajauttamiseksi sodanajan ympäristöön taktisen tasan johtamisjärjestelmien suorituskykyä ajatellen.

Liityntäsovelluksen ja uuden johtamisjärjestelmän DDS-toteutuksen valinnaksi tuli OpenSplice DDS. Merkittävimpinä tekijöinä tuotteen valintaan olivat toimintojen kattavuus standardin näkökulmasta, tuotteen pitkä historia ja tuotteen referenssit. DDS teknologiaan perustuvaa OpenSplice DDS tuotetta käytetään eri maiden laivastoissa taktisen tasan järjestelmissä sekä seuraavan sukupolven lennonohjausjärjestelmissä Euroopassa (Prismtech, 2008). Valintaan vaikutti myös tuotteen ilmaisuus ja sen avoimuus, koska kyseessä on vapaan lähdekoodin tuote LGPLv3 lisenssillä.

Varsinainen toteutustyö toteutettiin suuremman projektin osana. Liityntäsovelluksen toteuttaminen osaksi nykyistä järjestelmää onnistui valmiin sovelluskehyksen ansiosta hienosti. Sovelluskehys tarjosi valmiin konseptin ilmatilannekuvan tilaamiseen, sovelluksen hallintaan ja sen tapahtumien monitoroimiseen. DDS osuuden tekeminen aiheutti enemmän ongelmia, kuten oli etukäteen odotettavissakin.

Liityntäsovelluksella olevilla toiminnallisuuksilla pystytään tuottamaan nykyisen johtamisjärjestelmän ilmatilannekuvan tiedot seurannan tunnistustietoineen ja paikkatietoineen uuden järjestelmän käyttöön DDS-teknologialla. Liityntäsovellus ylläpitää ilmatilannekuvaa reaaliaikaisesti julkaisija–tilaaja–mallia hyödyntäen siten, että molemmat johtamisjärjestelmät näkevät saman tilannekuvan samanaikaisesti. OpenSplice DDS -tuotteen tarjoamat C++- ja Java-ohjelmointirajapinnat mahdollistivat hyvin kahden eri sukupolven johtamisjärjestelmien liittämisen yhteen, avoimia standardeja hyödyntäen.

Yleisesti voidaan todeta, että DDS tulee olemaan tulevaisuudessa varteenotettava välikerrosarkkitehtuurin vaihtoehto reaaliaikaisissa tiedonhajautuksen järjestelmissä. DDS standardin kehittyessä tulevat siihen sitoutuneiden tuotteiden ominaisuudet paranemaan. Avoimen lähdekoodin mahdollistava yhteisöllisyys ja sen mukana tuomat uudet ominaisuudet ja mielenkiintoiset ratkaisut tulevat näyttämään mihin suuntaan DDS:n tulevaisuus on menossa.

LÄHTEET

Bond, R.A., Martinez, D.R. & Vai, M.M., 2008. *High Performance Embedded Computing Handbook: A Systems Perspective*. Taylor & Francis Group. 376 s.

Buschmann, F. et al., 1996. *Pattern - Oriented Software Architecture: A System of Patterns*. John Wiley & Sons. 467 s.

Corsaro, A., 2009. *simd-cxx*. Saatavilla: http://code.google.com/p/simd-cxx/ [Viitattu 21 Tammikuu 2010].

Corsaro, A. et al., 2006. Quality of Service in Publish/Subscribe Middleware.

Field, D., 2008. *The Impact of Multi-core Processors on Application*. Saatavilla: http://www.hpcwire.com/offthewire/17910484.html [Viitattu 15 Helmikuu 2009].

Frolov, A.U., DiPippo, L.C. & Fay-Wolfe, V., 2008. Real-Time Data Distribution. *Handbook of Real-Time and Embedded System*. Taylor & Francis Group. Luku 25.

Gamma, E., Helm, R., Johnson, R. & Vlissides, J., 1995. *Design Patterns: Elements of Reusable Object-Oriented Software*. Addison-Wesley. 395 s.

Hayes, R.E. & Alberts, D.S., 2006. ISBN 1-893723-17-8 *Understanding command and control*. DoD Command and Control Research Program.

IEEE Computer Society, 2006. Virtual Bridged Local Area Networks. *Std 802.1Q-2005 IEEE Standard for Local and metropolitan area networks*. 285 s.

Ilmavoimat, 2004. *Tiedotteet*. Saatavilla: http:/www.mil.fi/ilmavoimat/tiedotteet/776.dsp [Viitattu 25 Tammikuu 2009].

Ilmavoimat, 2009. *Perustietoa*. Saatavilla: http://www.ilmavoimat.fi/ [Viitattu 15 Tammikuu 2009].

Insta DefSec Oy, 2006. *Finnish Air Defence Command And Control System*. 47 s.

Jussila, A., 2004. *Ilmatorjunta: Keskitettyä tulenkäytön johtamista 60-vuotta*. Saatavilla: http://www.ilmatorjuntaupseeriyhdistys.fi/3_2004/tekstit/tkjoht.htm [Viitattu 25 maaliskuu 2009].

Laplante, P.A., 2004. *Real-Time Systems Design and Analysis*. Kolmas painos ed. Institute of Electrical and Electronics Engineers. 528 s.

OMG, 2004. *Common Object Request Broker Architecture: Core Specification Version 3.0.3*.

OMG, 2007. *Data Distribution Service for Real-time Systems Version 1.2*. 248 s.

OMG, 2008. *The Real-time Publish-Subscribe Wire Protocol DDS Interoperability Wire Protocol Specification Version 2.1*. 200 s.

OMG, 2009. *DDS Interoperability Demo*. Saatavilla: http://www.omg.org/news/meetings/GOV-WS/pr/rte-pres/ddsi-demo.pdf [Viitattu 21 Tammikuu 2010].

OMG, 2009. *mars/09-12-16 (Java 5 Language PSM for DDS RFP)*. Object Management Group.

OpenSplice, 2009. *OpenSplice DDS Community*. Saatavilla: http://www.opensplice.org [Viitattu 17 Tammikuu 2010].

Peltola, T., 2008. *Diplomityö: Reaaliaikavälikerroksen arviointi johtamisjärjestelmäkäyttöön*. 57 s.

Prismtech, 2008. *OpenSplice DDS in Defense & Aerospace*. Saatavilla: http://www.opensplice.com/public-upload/OpenSpliceDDS_DAv20.pdf [Viitattu 19 Helmikuu 2010].

PrismTech, 2009. *OpenSplice DDS version 4.3 Deployment Guide*. PrismTech.

Puolustusvoimat, 2002. *Sotilaallinen maanpuolustus*. Helsinki: Pääesikunnan tiedotusosasto.

Puolustusvoimien Teknillinen Tutkimuslaitos, 2008. ISBN 978-951-25-1891-3 *Sotatekninen arvio ja ennuste 2025*. Ylöjärvi: Puolustusvoimien Teknillinen Tutkimuslaitos.

Ruotuväki, 2006. 44(967).

Schmidth, D.C., Corsaro, A. & van't Hag, H., 2008. Addressing the Challenges of Tactical Information Management in Net-Centric System with DDS. *Software Engineering Technology*, 2008(March), s. 24 - 29.

Sun Microsystems, 2002. *Java Message Service Specification - version 1.1*. Saatavilla: http://java.sun.com/products/jms/docs.html [Viitattu 26 maaliskuu 2009].

Thales, 2004. *PrismTech to market Thales Nederland SPLICE-DDS Data Distribution Service for Real-time Systems*. Saatavilla: http://www.thales-nederland.nl/nl/news/archive/2004/nov10-2004.shtml [Viitattu 21 Tammikuu 2010].

Valtioneuvoston selonteko, 2009. *Suomen turvallisuus- ja puolustuspolitiikka 2009*.

Yleisradio, 2008. *Venäjä tunnusti viimeinkin ilmatilan loukkauksen*. Saatavilla: http://www.yle.fi/ [Viitattu 20 Tammikuu 2009].

LIITE 1 Viitteelinen IDL-kuvaus

```
module c2
{
  struct metatiedot
  {
    <<metatiedot>>;
  };

  struct TrackPosition
  {
    long id;
    metatiedot m;

    double wgs84lat;
    double wgs84lon;
    double nopeus;
    double suunta;
  };
#pragma keylist TrackPosition id

  struct Track
    {
      long id;
      metatiedot m;

      <<tunnistetiedot>>;
    };
#pragma keylist Track id
};
```

LIITE 2 URI-konfiguraatio unicast-tiedonsiirrolle

```
<OpenSplice>
  <Domain>
    <Name>unicast_test</Name>
    <Database>
      <Size>10485760</Size>
    </Database>
    <Lease>
      <ExpiryTime update_factor="0.05">60.0</ExpiryTime>
    </Lease>
    <Service name="networking">
      <Command>networking</Command>
    </Service>
    <Service name="durability">
      <Command>durability</Command>
    </Service>
  </Domain>
  <NetworkService name="networking">
    <Partitioning>
      <GlobalPartition Address="225.0.0.1, 172.20.60.1, 172.20.62.1"/>
    </Partitioning>
    <Channels>
      <Channel default="true" enabled="true" name="BestEffort"
reliable="false">
        <PortNr>53340</PortNr>
        <Sending>
          <DontRoute>0</DontRoute>
          <TimeToLive>64</TimeToLive>
```

```
        </Sending>
      </Channel>
      <Channel enabled="true" name="Reliable" reliable="true">
        <PortNr>53350</PortNr>
        <Sending>
          <DontRoute>0</DontRoute>
          <TimeToLive>64</TimeToLive>
        </Sending>
      </Channel>
    </Channels>
    <Discovery enabled="true">
      <PortNr>53360</PortNr>
        <Sending>
          <Interval>200</Interval>
          <DontRoute>0</DontRoute>
          <TimeToLive>64</TimeToLive>
        </Sending>
    </Discovery>
    <General>
      <NetworkInterfaceAddress>eth0</NetworkInterfaceAddress>
      <Reconnection allowed="true"/>
    </General>
  </NetworkService>
  <DurabilityService name="durability">
    <Network>
      <InitialDiscoveryPeriod>2.0</InitialDiscoveryPeriod>
      <Alignment>
        <TimeAlignment>FALSE</TimeAlignment>
```

```
        <RequestCombinePeriod>
          <Initial>2.5</Initial>
          <Operational>0.1</Operational>
        </RequestCombinePeriod>
      </Alignment>
      <WaitForAttachment maxWaitCount="10">
        <ServiceName>networking</ServiceName>
      </WaitForAttachment>
    </Network>
    <NameSpaces>
      <NameSpace alignmentKind="Initial_and_Aligner"
durabilityKind="Durable">
        <Partition>*</Partition>
      </NameSpace>
    </NameSpaces>
  </DurabilityService>
</OpenSplice>
```