KRIITTISEN INFRASTRUKTUURIN TILANNEKUVAJÄRJESTELMÄ

LAURI LÄÄPERI, LAURI RUMMUKAINEN JA JOUKO VANKKA

Lauri Lääperi on tutkija Maanpuolustuskorkeakoulun Sotatekniikan laitoksella

Lauri Rummukainen on tutkija Maanpuolustuskorkeakoulun Sotatekniikan laitoksella

Jouko Vankka on sotatekniikan professori Maanpuolustuskorkeakoulun Sotatekniikan laitoksella

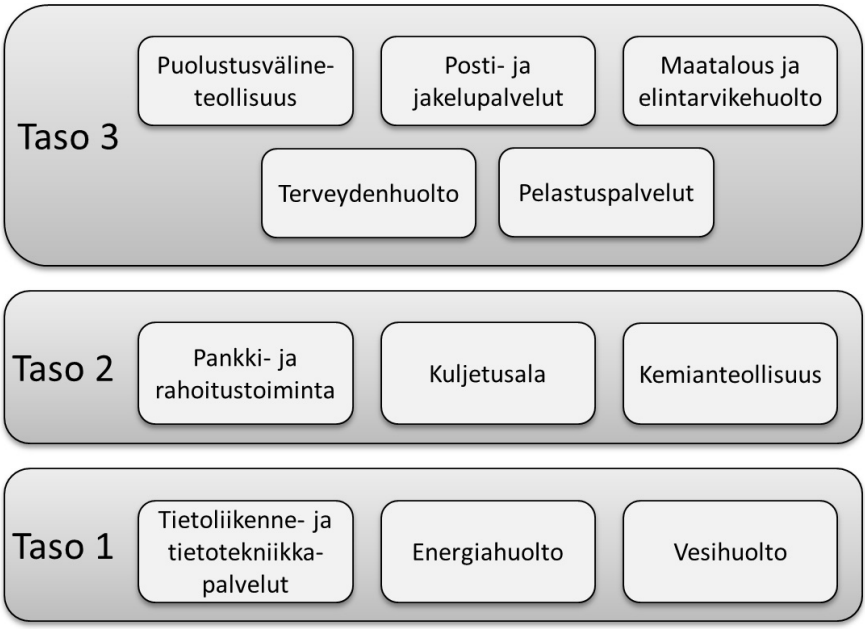
Tämä artikkeli on tehty osana Tekesin rahoittamaa DiSCI- (Digital Security of Critical Infrastructures) tutkimushanketta, jossa tutkitaan ratkaisuja suomalaisen kriittisen infrastruktuurin turvallisuuden parantamiseksi. Hankkeessa on tutkimuspartnereina Maanpuolustuskorkeakoulun lisäksi Aalto-yliopisto ja Stonesoft Oyj (nykyisin osa Intel/McAfee konsernia).

JOHDANTO

Moderni yhteiskunta on lähes täysin riippuvainen useiden palveluiden, kuten sähkön, veden ja tietoliikenteen, keskeytyksettömästä tarjonnasta. Yhteiskunnalle kriittisiä palveluita tuottavia toimialoja kutsutaan yhdessä kriittiseksi infrastruktuuriksi ja niiden turvaaminen on keskeistä elinolojen ylläpitämiseksi. Kriittisen infrastruktuurin turvaamisessa nopea häiriötilanteista palautuminen on erityisen tärkeää, jotta palvelukatkosten vaikutukset kyetään pitämään mahdollisimman pieninä. Nopea ja tehokas reagointi kokonaistilanteen korjaamiseksi vaatii yhteistä tilannekuvaa, jota eri toimijat voivat hyödyntää oman toimintansa ohjaamisessa ja koordinoitujen päätösten tekemisessä. Kriittinen infrastruktuuri sisältää monia riippuvuussuhteita, jonka vuoksi tilannetietoisuuden aikaansaaminen päätöksentekijöille on välttämätöntä nykytilan ja tulevaisuuden ymmärtämiseksi.

KRIITTINEN INFRASTRUKTUURI

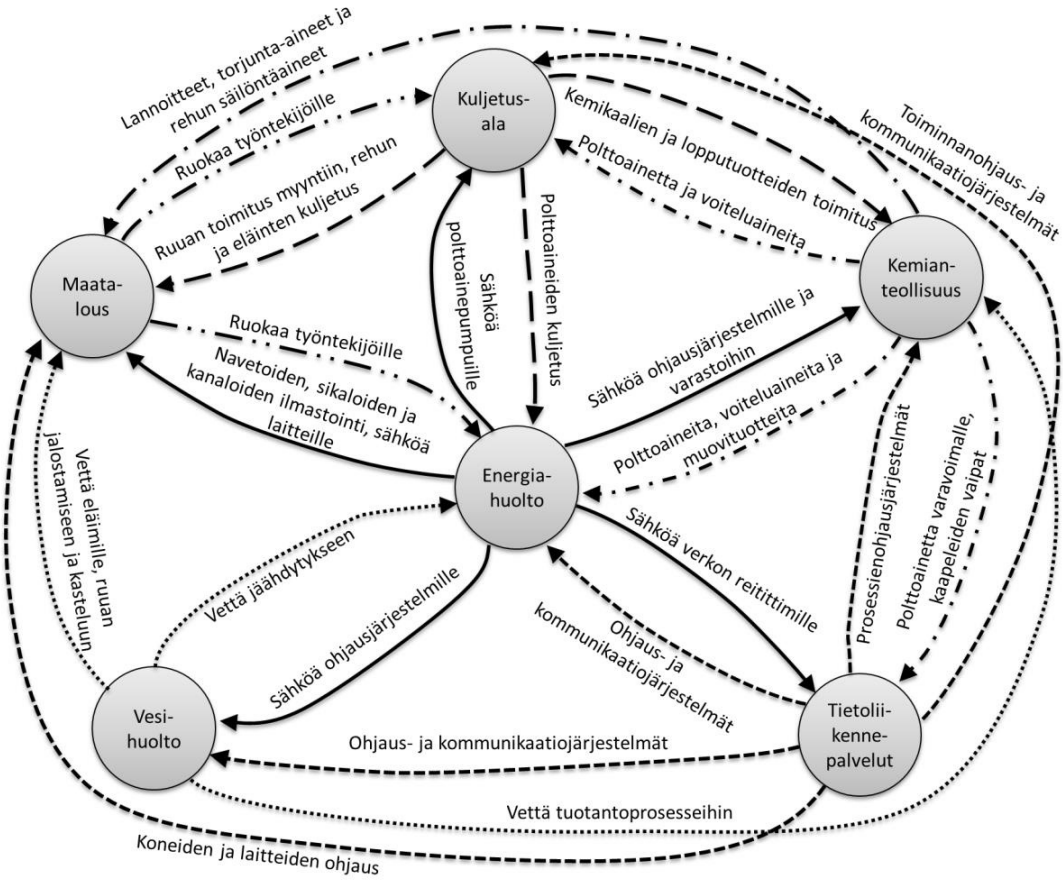
Yhdysvaltalainen Ted Lewis on määritellyt kriittisen infrastruktuurin koostuvan 11 eri sektorista, jotka ovat energiahuolto, vesihuolto, tietoliikenne- ja tietotekniikkapalvelut, pankki- ja rahoitustoiminta, kuljetusala, kemianteollisuus, puolustusvälineteollisuus, posti- ja jakelupalvelut, maatalous ja elintarvikehuolto, terveydenhuolto ja pelastuspalvelut [1]. Vaikka Lewisin määrittelemä kriittisen infrastruktuurin koostumus pohjautuu Yhdysvaltojen kriittiseen infrastruktuuriin, soveltuu määrittely myös muiden kehittyneiden maiden, kuten Suomen, kriittisen infrastruktuurin kuvaamiseen.



Kuva 1: Lewisin luokittelu kriittisen infrastruktuurin palveluille [1].

Lewis ryhmittelee lisäksi mainitut 11 sektoria kolmelle kriittisyystasolle kuvan 1 mukaisesti. Kriittisyystasot kuvaavat eri järjestelmien tukeutumista toisiinsa siten, että ylemmät tasot ovat riippuvaisia alempien tasojen toiminnasta. Alimman tason sektoreihin kuuluu energia- ja vesihuolto sekä tietoliikenne- ja tietotekniikkapalvelut, jotka yhdessä muodostavat pohjan koko yhteiskunnan toiminnalle. Lähtökohtana voidaan ensisijaisesti siis pitää näiden kolmen sektorin turvaamista.

Kriittisen infrastruktuurin kuvaaminen ei todellisuudessa onnistu kolmiportaisella riippuvuusmäärittelyllä. Nykyyhteiskunnan verkottuneesta kriittisestä infrastruktuurista voidaan havaita erittäin suuri määrä sekä yksi- että kaksisuuntaisia riippuvuussuhteita useiden eri sektoreiden välillä [2]. Kuva 2 esittää esimerkin kuuden eri toimialan muodostamasta riippuvuusverkosta, joka kuvastaa riippuvuussuhteiden hahmottaminen vaikeutta koko kriittisen infrastruktuurin ympäristössä. Huomionarvoinen asia on, että energiantuotannon lisäksi tietoliikenne- ja tietotekniikkapalvelut ovat nykyisin integroituneet lähes jokaiseen toimialaan. Palveluiden verkottuminen eri tietoliikennejärjestelmien kanssa on lisännyt huomattavasti riippuvuusverkon monimutkaisuutta ja on luonut uuden haavoittuvuuden myös kyberoperaatioiden muodossa.



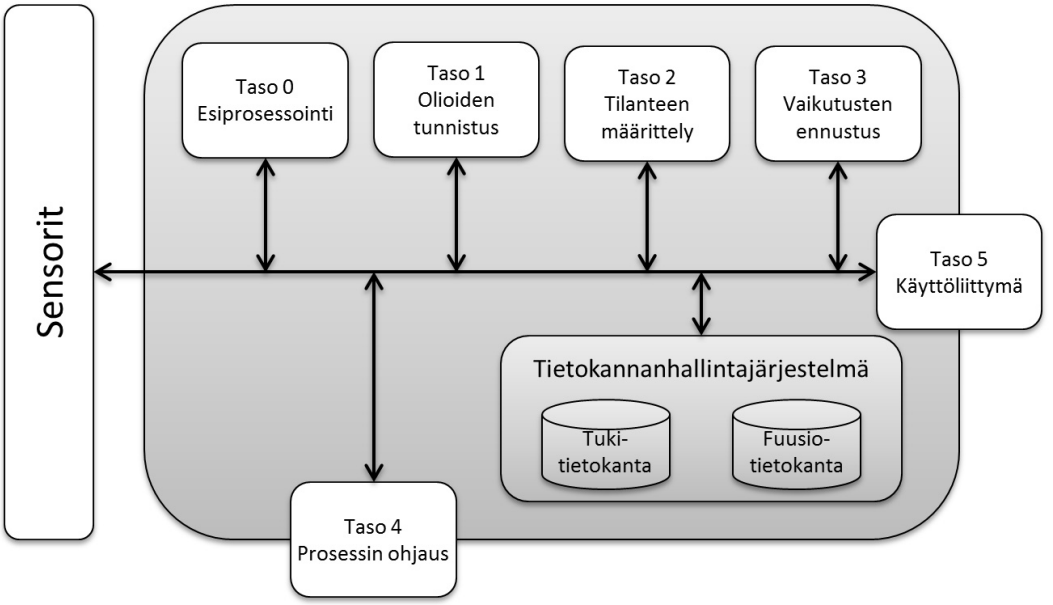
Kuva 2: Esimerkki kriittisen infrastruktuurin riippuvuuksista.

VAATIMUKSET

Kriittisen infrastruktuurin eri sektorit tulevat jatkuvasti yhä riippuvaisemmiksi toisistaan, kuten kuva 2 osoittaa. Jatkuvasti kehittyvät järjestelmät ja muuttuva ympäristö luovat suuria haasteita informaation keräämiselle ja integraatiolle. Tällaisen dynaamisen järjestelmän tuottaman tiedon tehokas hyödyntäminen vaatii, että järjestelmä itsessään on palvelukeskeinen [3]. Kohdejärjestelmien tuottaman informaation integraatio- ja muita vaatimuksia, kuten riippuvuussuhteiden selvittämistä, skaalautuvuutta ja visualisointia käsitellään tarkemmin seuraavaksi.

Integraatio

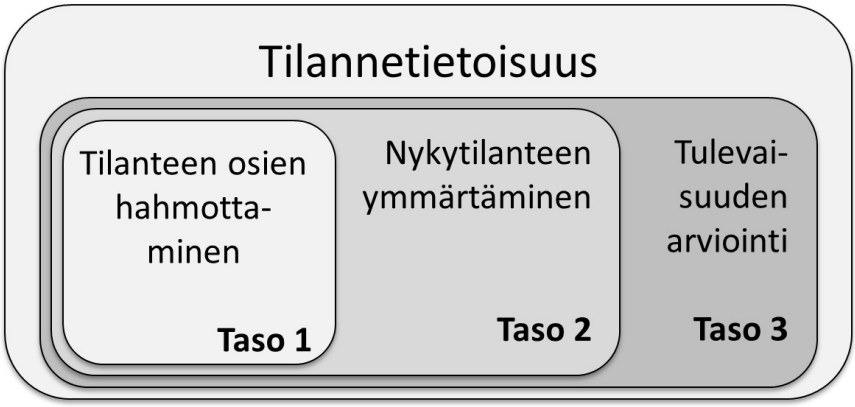
Erinäiset kriittisen infrastruktuurin kohdejärjestelmät tuottavat hyvin vaihtelevaa dataa niin rakenteeltaan kuin sisäl-



Kuva 3: JDL-datafuusiomalli sovellettuna tilannekuvaympäristöön.

löltään. Joint Directors of Laboratories ryhmän kehittämä JDL-datafuusiomalli luo hyvän perustan toimimiseen kriittisen infrastruktuurin muodostamassa dynaamisessa ympäristössä. Tästä syystä JDL-mallin tarjoamaa prosessimallia käytetään tilannetietojärjestelmän vaatiman dataintegraation perustana. Prosessi yhdistää useista lähteistä kerätyn tiedon siten, että tarkkailtava tilanne voidaan ymmärtää paremmin [4]. Giacobe [5] hyödyntää JDL datafuusiomallia kyberturvallisuuden viitekehyksessä ja kuvaa sensorien tuottaman datan integraation kuusitasoisella fuusiomallilla, joka pitää sisällään kohdejärjestelmän esiprosessoinnin, olioiden tunnistuksen, tilanteen määrittelyn, vaikutusten ennustuksen, prosessin ohjaamisen ja käyttöliittymän (Kuva 3).

Kuvan 3 sensorit koostuvat tilannekuvajärjestelmän asiayhteydessä erilaisista kriittisen infrastruktuurin kohdejärjestelmistä kuten tietoverkoista ja ohjausjärjestelmistä. Fuusiomallin taso 0 hoitaa kohdejärjestelmän tuottaman raa'an datan muuttamisen muotoon, jota voidaan prosessoida yhteismitallisesti. Taso 1 kerää tason 0 tuottaman datan ja tunnistaa yksittäiset oliot kohdejärjestelmien tuottamasta datavirrasta. Tasolla 2 määritetään nykyhetken tilanne, jota täydennetään tason 3 tuottamalla vaikutusennusteella. Taso 4 ohjaa fuusioprosessin toimintaa esimerkiksi mukauttamalla lähdedatan prosessointia tai yksittäisten tapahtumien tunnistamista. Tasolla 5 fuusion tulos esitetään käyttäjälle siten, että hänen tilannetietoisuutensa tarkkailtavasta järjestelmää



Kuva 4: Endsleyn tilannetietoisuuden malli [6].

paranee. Käyttökelpoisen ja tilannetietoisuutta parantavan tilannekuvan muodostamiseen tarvitaan kaikki mainitut kuusi datafuusion tasoa [5].

JDL-malli sopii siitäkin syystä tilannekuvajärjestelmän perustaksi, että se tukee hyvin kuvassa 4 esitettyä Mica Endsleyn kehittämää tilannetietoisuuden mallia. Endsleyn malli koostuu kolmesta tasosta, jotka koskevat tilanteen eri osien havaitsemista, nykytilanteen ymmärtämistä ja tulevaisuuden tilanteen arviointia. Nämä samat osa-alueet löytyvät myös kuvan 3 esittämän JDL-mallin tasoilta 1-3. Endsleyn malleja voidaan soveltaa suoraan tilannekuvajärjestelmän käyttöliittymän suunnitteluun, koska järjestelmän päämääränä on nimenomaan parantaa käyttäjien tilannetietoisuutta.

Riippuvuudet

Tilannekuvajärjestelmälle on olennaista kyetä tunnistamaan ja esittämään kriittisen infrastruktuurin sisäisiä riippuvuuksia. Kriittisestä infrastruktuurista voidaan tunnistaa neljä eri riippuvuustyyppiä: fyysinen, kyber, maantieteellinen ja looginen [2]. Mainitut riippuvuustyypit muuttuvat jatkuvasti kriittisen infrastruktuurin kehittyessä, mutta ne ovat lyhyellä aikajänteellä muuttumattomia vaatiessaan lähes aina ihmisten tekemiä sopimuksia tai päätöksiä, ks. Kuva 2. Lyhyen aikavälin staattisesta luonteesta johtuen riippuvuuksia ei ole syytä tarkastella samalla aikajänteellä kuin kohdejärjestelmien tuottamaa dataa. Tästä syystä riippuvuustiedon kerääminen tulisi hoitaa irrallaan kohdejärjestelmien tuottamasta datavirrasta.

Yhteisen riippuvuustietokannan ylläpitäminen on välttämätöntä, jotta fuusioprosessi pystyy rakentamaan riippuvuusverkon infrastruktuurin mallintamista varten. Riippuvuustietojen keskitetty tallennus tuo mukanaan huomioon otettavia asioita, kuten tiedon näkyvyyden rajoittamisen ja turvaamisen. Kriittisen

infrastruktuurin toiminnan kannalta riippuvuustietojen salassapito ulkopuolisilta tahoilta on tärkeää, sillä väärissä käsissä sen avulla voidaan helposti tunnistaa yhteiskunnan kannalta kriittiset solmut, joihin voidaan kohdistaa esimerkiksi kyberoperaatioita. On lisäksi huomioitava, että eri toimijoiden pääsyä kaikkien riippuvuussuhteiden havainnointiin tulisi rajoittaa, jotta yksityisten yritysten toimintaa ei vahingoiteta.

Skaalautuvuus

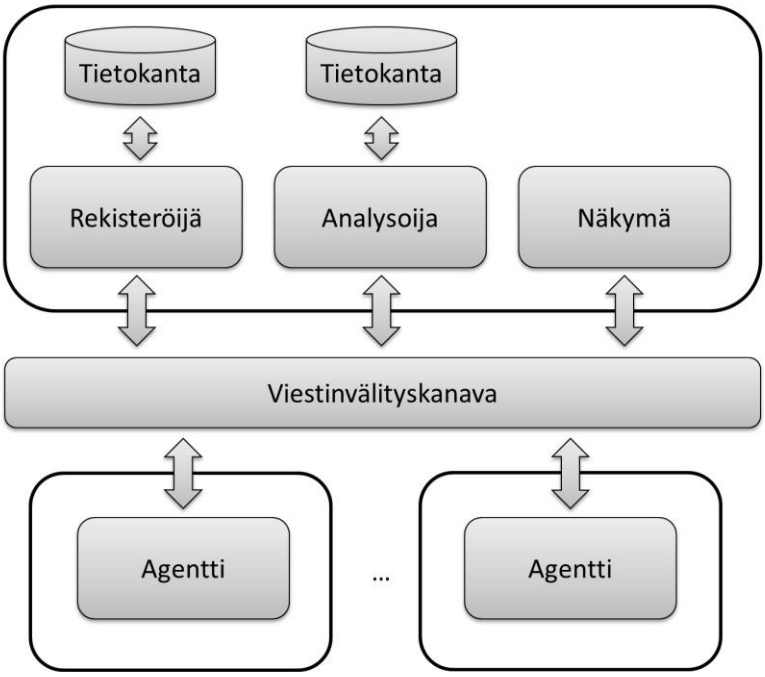
Tilannekuvajärjestelmä on tarkoitus toteuttaa kansallisella tasolla, joten sen on mahdollistettava datan keräys suuresta määrästä eri kohdejärjestelmiä. Tästä syystä uusien tiedonlähteiden lisäys järjestelmään tulee vaatia vain vähän resursseja. Lisäksi järjestelmän ydinosien tulee olla hajautettavissa, jotta laskentaresurssit voidaan jakaa usean palvelimen kesken. Etenkin JDL-mallin taso 1 (Kuva 3), joka vastaanottaa kohdejärjestelmistä saapuvan datan, tulee kyetä toimimaan rinnakkaisten palvelinten välillä.

Kuvan 3 esittämä JDL-malli tukee hyvin järjestelmän skaalautuvuutta tarjoamalla prosesseja, jotka kykenevät yhdistämään ja suodattamaan eri järjestelmistä kerättyä dataa. Esimerkiksi tason 0 avulla voidaan hyvin erilaisten kohdejärjestelmien integraatio toteuttaa järjestelmäkohtaisella esiprosessointi komponentilla. Tällöin taso 0 toimii rajapintana kahden eri järjestelmän välillä ja tarjoaa mahdollisuuden kerätä ja välittää eteenpäin dataa, joka on tilannekuvan kannalta oleellista. Joustavan integraation lisäksi taso 0 toimii ensimmäisenä suodattimena tilannekuvajärjestelmään saapuvan datan rajoittamisessa. Kohdejärjestelmän esiprosessoinnilla voidaan tehokkaasti rajoittaa seuraaville tasoille suunnatun datan määrää jättämällä kohdejärjestelmän tilaan vaikuttamattomat viestit välittämättä eteenpäin. Myös JDL-mallin seuraavat tasot, etenkin taso 1, pystyvät yhdistämään ja suodattamaan ylimääräistä dataa, jotta tilannekuvajärjestelmän ydinosat eivät ruuhkaudu.

Visualisointi

Tilannekuvajärjestelmän tärkein tehtävä on parantaa eri päätöksentekijöiden tilannetietoisuutta. Useat käyttäjät ja ympäristöt luovat monia haasteita tilannekuvan esitystapojen toteuttamiseen. Esimerkiksi kriittisen infrastruktuurin toimijoilla on hyvin erilaiset lähestymistavat ja prioriteetit järjestelmien turvaamisesta ja tiedon jakamisesta toisten toimijoiden kanssa. Lisäksi eri päätöksentekijöiden tiedon tarve vaihtelee yksittäisen toimijan toimintaa koskevista päätöksistä aina virkavallan vaatimaan laajemman kokonaisuuden hahmottamiseen. Siksi tilannekuvaa on pystyttävä tarjoamaan useilla eri tasoilla tukien niin alhaisen kuin korkean tason päätöksentekoa.

Eri esitystasojen lisäksi tulee tiedon näkyvyyttä rajoittaa eri päätöksentekijöiden välillä. Esimerkiksi virkavalta tarvitsee tietoa useiden toimijoiden välisistä riippuvuuksista, kun taas yksittäiselle toimijalle riittää, että hän saa tiedon niistä toimijoista, joista hän itse on riippuvainen. Visualisoinnissa suureen rooliin nousee myös käyttäjäryhmän heterogee-



Kuva 5: Arkkitehtuurin komponentit.

nisuus. Jotta hyvin erilaiset käyttäjät pystyvät hyödyntämään tilannekuvaa, on se kyettävä esittämään erilaisilla teknisillä tasoilla. Korkean tason päättäjien on hahmotettava erinäisten riippuvuussuhteiden aiheuttamat vuorovaikutukset, kun taas yhden toimijan tilannekuvan tulisi tarjota tarkempaa tietoa omaan järjestelmään vaikuttavien toimijoiden tilasta ja niissä tapahtuvista muutoksista.

ARKKITEHTUURI

Edellä esitellyt vaatimukset ohjaavat vahvasti tilannekuvajärjestelmän arkkitehtuurin suunnittelua. Järjestelmän on täytettävä vaatimukset, jotta se pystyisi toimimaan kriittisen infrastruktuurin muodostamassa ympäristössä ja tarjoamaan hyödyllistä tilannekuvaa eri toimijoille. Järjestelmän tulee olla joustava ja mukautuva alati kehittyvässä ympäristössä, joten sen pitää olla rakenteeltaan palvelukeskeinen [3]. Järjestelmän tulee siten koostua lähes itsenäisistä komponenteista, jotka tarjoavat yksittäisiä palveluita tilannekuvan tuottamiseksi. Eri komponenttien tulee tarjota selkeät rajapinnat, joiden kautta muut osat voivat käyttää niiden tarjoamia palveluita.

Kuvassa 5 esitetään tilannekuvajärjestelmän korkean tason arkkitehtuuri, joka pohjautuu kuvan 3 JDL-malliin. Palvelukeskeisen järjestelmän keskeiseksi kom-

ponentiksi muodostuu kuvassa 5 esitetty yhteinen viestinvälityskanava, joka mahdollistaa eri komponenttien saumattoman kommunikoinnin. Kohdejärjestelmien integraatio ja esiprosessointi JDL-mallin tasolla 0 (Kuva 3) voidaan suorittaa järjestelmäkohtaisella agentti-komponentilla, joka kykenee keräämään kustakin kohdejärjestelmästä tietoa. Agenttien hallintaan ja tunnistamiseen tarvitaan lisäksi rekisteröijä, joka on vastuussa myös riippuvuustiedon keräämisestä. Järjestelmien tuottaman tiedon analysoinnin JDL-mallin tasoilla 1-3 hoitaa analysoija-komponentti, jonka tulos esitetään päätöksentekijöille näkymä-komponentin kautta. Rekisteröijä ja analysoija tarvitsevat omat tietokantansa agenttien ja käyttäjien tallentamiseen sekä datafuusioprosessin suorittamiseen.

Agentti

Kriittisen infrastruktuurin kohdejärjestelmien on kyettävä tuottamaan tietoa tilannekuvajärjestelmään. Eri kohdejärjestelmien tuottaman raa'an datan integrointi suoraan ei ole toimiva ratkaisu, koska yksittäisellä taholla ei ole kykyä ymmärtää kaikkia kohdejärjestelmiä ja tunnistaa niiden virheellistä toimintaa. Lisäksi, useat toimijat ovat yksityisomistuksessa, eikä heidän järjestelmiinsä ole suoraa pääsyä. Ongelman muodostaa myös tietosuoja yritykselle arkaluontoisen tiedon jakamisessa. Vaikka esimerkiksi teollisuuden automaatiojärjestelmiä toimittavat vain muutamat valmistajat ja niitä käytetään pitkälti samalla tavalla eri toimijoiden puolesta [7], on järjestelmien kirjo koko kriittisen infrastruktuurin tapauksessa liian suuri suoraan integraatioon. Rajoituksista johtuen JDL-mallin tason 0 esiprosessointi on tarpeellinen ja se voidaan suorittaa järjestelmäkohtaisella agenttiohjelmalla.

Agenttipohjainen lähestymistapa sopii hyvin kriittisen infrastruktuurin ympäristöön. Agentin tehtävä on kerätä ja tunnistaa oleellinen tieto järjestelmästä ja sen tilasta. Lisäksi se pystyy muokkaamaan datan yhteismitalliseen muotoon. Agenttilähestymistavan hyvänä puolena on myös eri kohdejärjestelmiä ylläpitävien ammattilaisten sitouttaminen. Ylläpitäjien ammattitaito saadaan hyödynnettyä jo datafuusioprosessin ensimmäisellä tasolla, koska he itse muokkaavat agentin tuottamaan tietoa omasta järjestelmästään.

Agentin havaitsemat muutokset kohdejärjestelmässä välitetään viestinvälityskanavan (Kuva 5) kautta tapahtumina (event). Tapahtuman tietomuoto on yhteinen kaikille agenteille ja pitää sisällään mm. tapahtuman luokittelun ja vakavuuden. Eri kohdejärjestelmät eroavat huomattavasti toisistaan ja tuottavat siten hyvin erilaista dataa, joten tapahtuman tietomuodon on oltava joustava ja sallittava monenlaisen tiedon välityksen. Lisäksi tietomuodon tulisi olla helposti laajennettavissa, jotta se sopisi kriittisen infrastruktuurin jatkuvasti kehittyvään ympäristöön. Sopiva tietomuoto tapahtumalle on vapaamuotoinen avain-arvo pari lista, jossa muutamat avain parametrit on yhteisesti määritelty, mutta muiden parametrien käyttöä ei rajoiteta. Tällöin järjestelmät voivat tuottaa määriteltyjen parametrien lisäksi juuri sellaista tietoa kuin kykenevät.

Viestinvälittäjä

Palvelupohjainen rakenne ja JDL-datafuusiomalli tarvitsevat kaikkien järjestelmän komponenttien yhteistoiminnan mahdollistavaa tehokasta viestinvälityskanavaa, ks. kuva 5. Yhteinen viestinvälityskanava on erittäin tärkeä koko yhteiskunnan laajuisessa toteutuksessa, jossa eri osat ovat hajautettuna useille palvelimille ja laitteille. Lisäksi viestinvälityskanavan on pystyttävä käsittelemään suuria määriä viestejä ja mahdollistaa niiden reititys useisiin määränpäihin.

Järjestelmän vaatima joustava viestinvälitys voidaan toteuttaa viestinvälittäjä arkkitehtuurilla, jossa yksi tai useampia palvelimia muodostavat yhdessä viestinvälityskanavan kuvan 5 komponenttien välille ja tarjoavat joustavan reitityksen monimuotoiselle datalle. Viestinvälittäjäpalvelu voidaan mieltää tietynlaiseksi pilvipalveluksi, jossa eri komponenttien tarvitsee vain tietää sisäänmenokanava ja vastaanottaja, jonka jälkeen viestien reititys, jonotus ja tallennus tapahtuvat automaattisesti. Viestinvälittäjä tukee hyvin viestin toimittamista niin yhdelle kuin useammallekin vastaanottajalle.

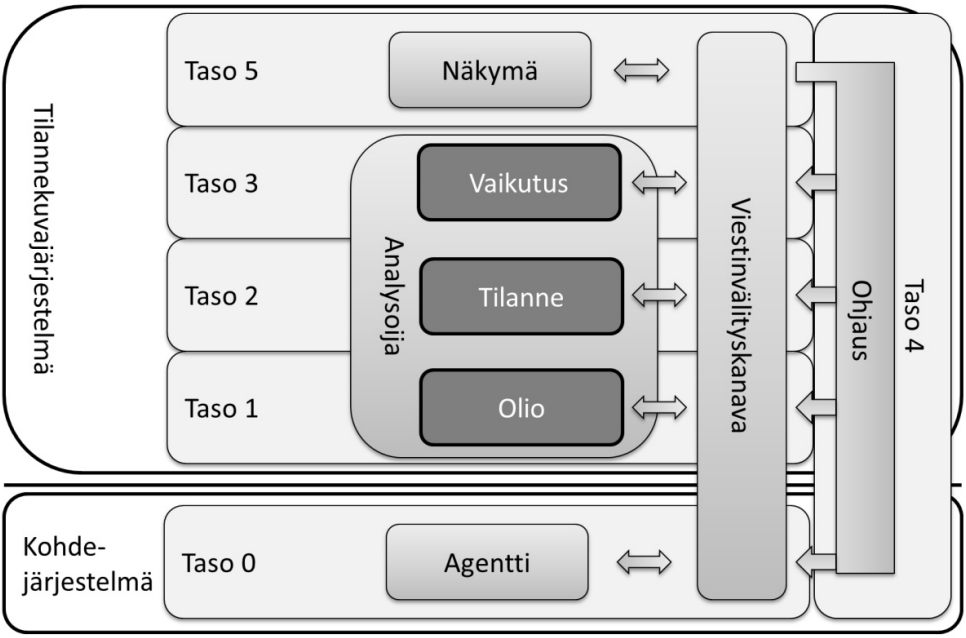
Rekisteröijä

Kuvan 5 rekisteröijän vastuulla on pääsääntöisesti agenttien tietojen rekisteröiminen järjestelmään ja sitä kautta riippuvuustiedon ylläpito. Käyttäjien erottelua varten rekisteröijä huolehtii myös heidän tunnistamisestaan ja valtuuttamisestaan muille kuvan 5 komponenteille. Kuten agenttien muokkaaminen kohdejärjestelmäkohtaisesti, myös niiden rekisteröinti, riippuvuuksien määrittely sekä päivitys on tehtävä kohdejärjestelmän asiantuntijoiden toimesta tai avustuksella. Agentit tulee rekisteröidä osaksi järjestelmää ja yksilöidä tunnisteella, jotta niiden tuottamat tapahtumat voidaan yhdistää fyysiseen järjestelmään. Koska agenttien mahdollinen määrä on hyvin suuri, on myös tunnisteavaruuden oltava riittävä tarjoamaan kaikille agenteille yksilölliset tunnisteet. Tunnisteet tulisi lisäksi jakaa mahdollisimman satunnaisesti, jottei niitä voida yhdistää tiettyihin kohdejärjestelmiin ilman rekisteröijän ylläpitämää tietoa.

Agenttia rekisteröitäessä tulee myös riippuvuustiedot syöttää rekisteröijän tietokantaan, jotta eri kohdejärjestelmien välisiä riippuvuuksia voidaan hyödyntää arvioitaessa tapahtumien vaikutuksia kriittisessä infrastruktuurissa. Asiantuntijat määrittävät mistä muista järjestelmistä he itse ovat riippuvaisia, millä tavalla, ja kuinka pitkään he pystyvät jatkamaan toimintaansa, mikäli riippuvuuden lähteenä oleva järjestelmä lakkaa toimimasta.

Analysoija

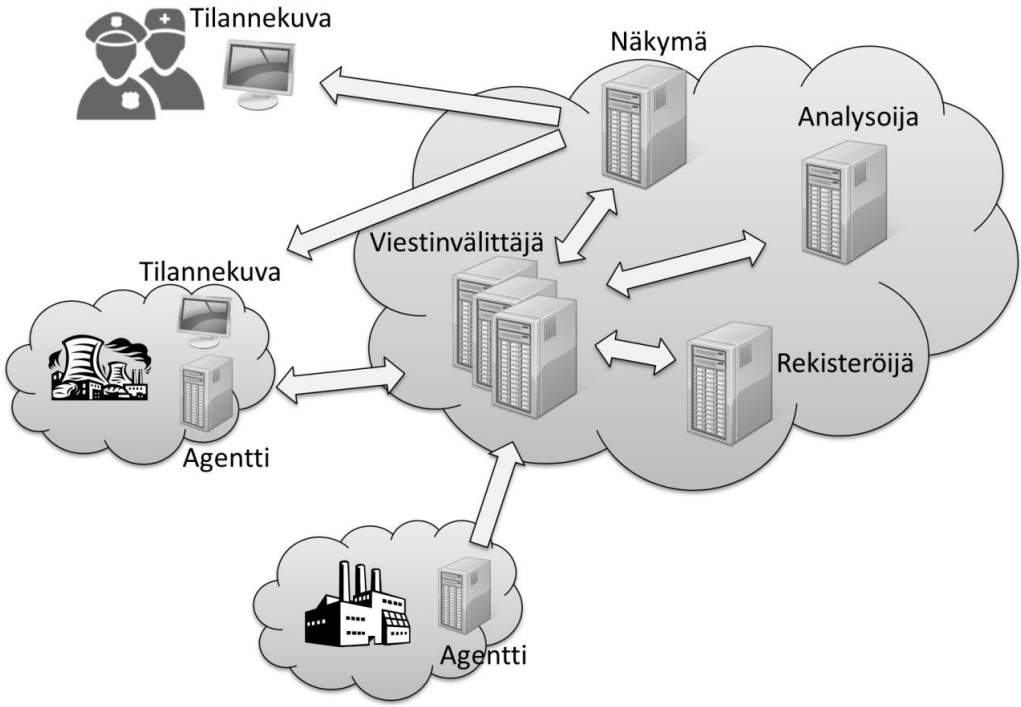
Tilannekuvajärjestelmän tärkeimpiä tehtäviä on analysoida kohdejärjestelmistä saatua dataa ja sen perusteella muodostaa yhtenäinen tilannekuva. Analyysi tuottaa tietoa eri JDL-mallin tasoilla (Kuva 3), joiden tuloksena tunnistetaan mukana olevat oliot, nykyinen tila ja vaikutus ennustus. Kuva 6 esittää arkkitehtuurin esittelemät komponentit, sekä analyysin sisällään pitämät osat suhteessa JDL-mallin tasoihin.



Kuva 6: Arkkitehtuuri suhteessa kuvan 3 JDL malliin.

Kuvan 6 olioanalyysi prosessoi agenttien tuottamia tapahtumia ja muodostaa niiden perusteella kuvan eri toimijoiden tilasta. Analyysin tässä osassa suodatetaan myös pois tapahtumat, jotka eivät kokonaiskuvan kannalta ole tarpeellisia. Myös uusia tapahtumia voidaan muodostaa, mikäli useista eri lähteistä saatava tieto viittaa jonkin suuremman tapahtuman olemassaolosta. Tällainen tapahtuma voisi olla esimerkiksi laajamittainen järjestelmien verkkoskannaus, joka kielii koordinoidusta kybertiedustelusta. Olioiden tunnistaminen tapahtuu pääosin tapahtumavirtoja analysoimalla, joten laskennassa voidaan hyödyntää monimutkaisten tapahtumien prosessointityökaluja (complex event processing) [8].

Kuvan 6 tilanne- ja vaikutusanalyysissa muodostetaan havaituista olioista nykyinen, eri kohdejärjestelmien muodostama nykytila, jota täydennetään riippuvuuksien avulla muodostetulla vaikutusennusteella. Tulevaisuuden ennustamisessa riippuvuussuhteiden kautta voidaan hyödyntää graafiteoreettisia menetelmiä laskemalla verkon kriittisiä solmuja ja ennustaa ongelmien leviämistä. Analyysin seurauksena voidaan luoda hälytyksiä varoittamaan toimijoita poikkeavista tilanteista ja antaa heille edellytykset ohjata omaa toimintaansa tilanteen mukaan.



Kuva 7: Esimerkki tilannekuvajärjestelmän rakenteesta.

Näkymä

Näkymäkomponentin tehtävä on tarjota käyttäjille tilannekuva sellaisessa muodossa, joka parhaiten parantaa kunkin päätöksentekijän tilannetietoisuutta. Käyttöliittymä tulisi tarjota joustavasti selainpohjaisena, jolloin sen käyttö onnistuu helposti tavallisilla tietokoneilla, valvomoissa tai jopa mobiililaitteilla. Selainpohjainen käyttöliittymä tekee siitä helposti käytettävän ja poistaa tarpeen erillisien ohjelmistojen asennukselle ja ylläpidolle. Selainkäyttöliittymällä on lisäksi helppo rajata tiedon näkyvyyttä ja tarjota erilaisia visualisointitapoja eri toimijoille.

TOTEUTUS

Järjestelmän keskeisen osat toteutettiin arkkitehtuurin toiminnallisuuden varmentamiseksi. Eri komponentit kehitettiin Spring-ohjelmistokehyksen [9] (framework) avulla, joka on avoimen lähdekoodin ohjelmistokehys Java-alustalle. Järjestelmän viestinvälittäjä toteutettiin Apache ActiveMQ [10] viestipalvelimen avulla, koska se tukee täysin Javan viestipalvelua (Java Message Service) ja yhdessä Apache Camel [11] ohjelmistokehyksen kanssa tukee etäkutsuttavia metodeja (Remote Method Invocation). Järjestelmän eri komponentit voivat tällöin kommunikoida hel-

posti keskenään tavallisten Java-rajapintojen kautta. Eri komponenttien vaatimat tietokantayhteydet toteutettiin Hibernate ORM [12] kirjaston avulla, joka mahdollistaa tietokannasta riippumattoman toteutuksen tekemisen. Kuva 7 esittää yleistason kuvan järjestelmän rakenteesta hajautettuna usealle palvelimelle ja toimijalle.

Agentti

Agentti-komponentin toteutus on järjestelmän käytettävyyden kannalta keskeistä, koska sen täytyy olla helposti laajennettavissa kohdejärjestelmäkohtaisesti. Mahdollisimman helppo muokattavuus saavutetaan, kun erotetaan järjestelmäkohtaiset ominaisuudet kaikille agenteille yhteisistä ominaisuuksista. Javalla toteutettu agentin perusosa antaa tuen tarvittaville ominaisuuksille kuten viestien lähettämiselle ja yhteydenpidolle tilannekuvajärjestelmän kanssa. Tällöin järjestelmän ammattilaisten tarvitsee toteuttaa ainoastaan pieni lisäkomponentti (plugin), joka kykenee keräämään järjestelmäkohtaisesti dataa ja luokittelemaan sitä. Agenttien erotteluun vaadittava tunniste saadaan rekisteröijältä kun agentti rekisteröidään osaksi järjestelmää.

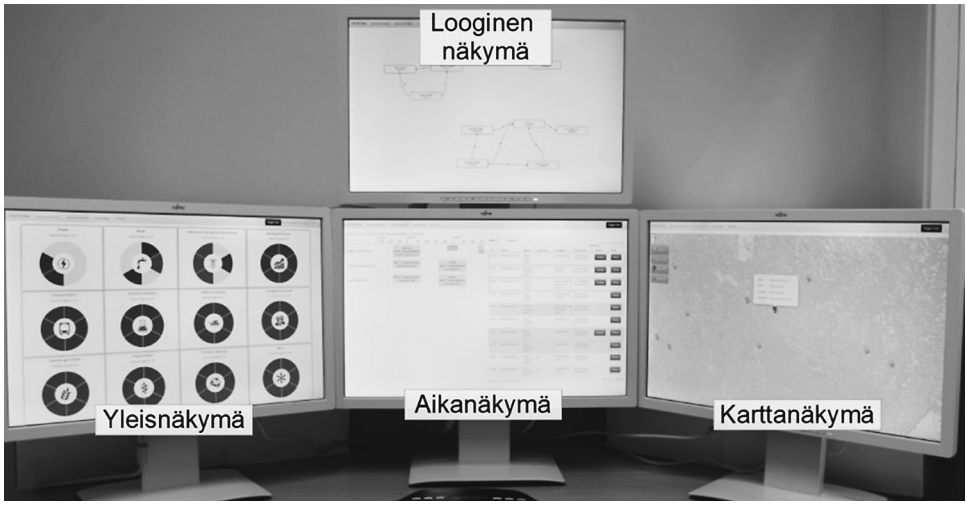
Viestinvälittäjä

ActiveMQ viestipalvelin mahdollistaa kaksi erilaista kommunikaatiokanavaa, otsikot ja jonot. Otsikko-kanava on eräänlainen yleislähetyskanava (broadcast), jossa kaikki sitä kuuntelevat tahot saavat saman viestin. Jono-kanavassa sen sijaan viestit välitetään aina yhdelle vastaanottajalle ja usean kuuntelijan tapauksessa kukin taho vastaanottaa vuorollaan viestejä. Tilannekuvajärjestelmässä eri kanavia voidaan hyödyntää tietyn kommunikaation toteuttamiseen. Esimerkiksi agenttien tuottamat tapahtumaviestit välitetään otsikko-tyyppisessä kanavassa, jolloin viestit voidaan toimittaa usealle järjestelmän komponentille yhtäaikaisesti, ks. Kuva 7.

Jonojen käyttö sen sijaan painottuu kuvan 5 komponenttien tarjoamien palveluiden suorittamiseen. Komponenttien tarjoamien rajapintojen käyttäminen jonojen kautta on perusteltua, koska tietty operaatio ja siihen mahdollisesti liittyvä vastaus on tarkoitus suorittaa ainoastaan kerran, vaikka palvelun tarjoajia olisi todellisuudessa useita. Joustavan viestinvälittäjän avulla myös itsenäisten komponenttien levittäminen useille eri palvelimille onnistuu helposti.

Rekisteröijä

Rekisteröijä tarjoaa käyttäjille oman käyttöliittymän, jonka kautta agentit voidaan rekisteröidä ja niiden tietoja päivittää. Agenttien tunnisteena käytetään UUID (universally unique identifier) tunnisteita, jotka luodaan satunnaisesti rekisteröinnin yhteydessä. Rekisteröidyt agentit liitetään aina jonkin käyttäjän hallintaan, jolloin voidaan hallita agenttien näkyvyyttä. Käyttäjille tarjottavan käyttöliittymän lisäksi rekisteröijä tarjoaa myös muiden komponenttien käytettäväksi rajapinnan, jonka kautta saadaan tehtyä kyselyitä mm. agenttien välisistä riippuvuuksista. Rekisteröijä toimii siten agentti- ja käyttäjätietokannan rajapintana.



Kuva 8: Toteutuksen käyttöliittymä, joka rakentuu neljästä näytöstä

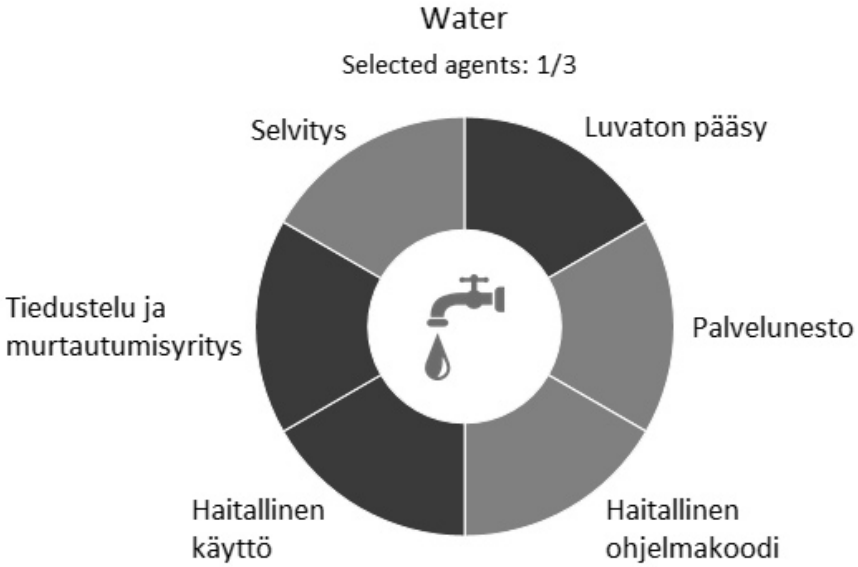
Analysoija

Kohdejärjestelmästä kerätyt tapahtumat analysoidaan kolmella eri komponentilla kuvan 6 mukaan. Analyysi perustuu JDL-malliin (Kuva 3) ja tuottaa tapahtumista olioita, järjestelmän nykytilan ja tapahtumien vaikutusarvion. Analyysiosien tulosten muodostamiseen hyödynnetään lisäksi rekisteröijältä saatavia tietoja agenteista ja niiden riippuvuuksista. Analysoijan eri sisäprosessit (olio, tilanne ja vaikutus) välittävät kukin omaan otsikko-tyyppiseen kanavaan prosessoinnin tuloksena syntyneet muutokset, jolloin useat muut komponentit voivat kuunnella analyysin tuloksia reaaliaikaisesti. Muutosten lisäksi analysoijan prosesseilta voidaan kysellä tietoja samaan tapaan kuin rekisteröijältäkin.

Näkymä

Kehitetty selainkäyttöliittymä tarjotaan eri käyttäjille näkymäkomponentin kautta. Suunniteltu käyttöliittymä koostuu neljästä näytöstä, jotka on aseteltu kuvan 8 mukaisesti: kolme näyttöä vierekkäin, ja neljäs näyttö keskimmäisen yläpuolella.

Vasemmanpuoleisimmassa näytössä esitetään käyttäjälle kriittisen infrastruktuurin nykytila helposti ymmärrettävässä muodossa. Näyttö koostuu kahdestatoista tilaympyrästä, jotka kuvaavat Lewiksen yhtätoista sektoria (Kuva 1) ja yhtä ylimääräistä sektoria muille toimijoille. Jokaisen sektorin kohdalla on kuusiosainen ympyrä (Kuva 9), jonka osuudet kuvaavat eri tapahtumaluokkia. Toteutuksessa käytössä olevat tapahtumaluokat ovat: luvaton pääsy, palvelunesto, haitallinen ohjelmakoodi, haitallinen käyttö, tiedustelu ja murtautumisyritys sekä selvi-



Kuva 9: Yleisnäkymän vesisektorin tilaympyrä tapahtumaluokkineen

tys. Yleisnäkymässä käyttäjä pystyy myös määrittämään, mitä toimijoita hän haluaa seurata. Näytön tarkoituksena on tarjota nopeasti informaatiota, toimivatko kaikki sektorit ongelmitta, ja jos ongelmia on ilmennyt, minkä tyyppisistä ongelmista on kyse. Yleistason tilannekuva on myös yksi tärkeä osatekijä Endsleyn tilannetietoisuusteoriassa [13].

Keskimmäinen näyttö (aikanäkymä) on tarkoitettu käyttäjän päänäkymäksi. Aikanäkymässä näytetään perinteisen tapahtumalokin kaltaisesti kaikki käyttöliittymälle lähetetyt tapahtumat. Jotta käyttäjä pystyy muodostamaan hyvän kuvan kriittisen infrastruktuurin nykytilasta, täytyy hänen ymmärtää jo ilmenneet tapahtumat, ja tarvittaessa etsiä uusiin tapahtumiin liittyviä aiempia tapahtumia. Siksi aikanäkymässä käyttäjälle tarjotaan myös aikajana (Kuva 10), jonka avulla tapahtumien järjestys eri toimijoiden kohdalla käy havainnollisesti ilmi.

Oikeinpuolimmaisin näyttö esittää tapahtumien ja toimijoiden maantieteellisen jakautumisen. Näytön avulla käyttäjä pystyy arvioimaan alueellisia tapahtumia ja niiden laatua. Karttanäkymä on esitetty kuvassa 11.

Neljäs näyttö on sijoitettu keskimmäisen näytön yläpuolelle. Tämän loogisen näkymän tarkoituksena on antaa käyttäjälle kuva laitteiden loogisesta sijainnista osana koko kriittistä infrastruktuuria. Looginen näkymä mahdollistaa myös eri tapahtumien vaikutuksien arvioimisen. Näkymä koostuu toimijasolmuista ja niiden välisistä riippuvuuksista (Kuva 12). Riippuvuudet on esitetty nuolina kuvaamaan kuka toimija riippuu kenestäkin,

ja niiden yhteydessä esitetään lukuarvo, joka kuvaa riippuvuuden kriittistä aikaa. Kriittinen aika kuvaa sitä aikaa, jonka riippuvainen kykenee toimimaan ilman toista toimijaa. Riippuvuudet on myös väritetty neljällä eri värillä kuvaamaan eri riippuvuustyyppejä: fyysistä, kyber, maantieteellistä tai loogista riippuvuutta.

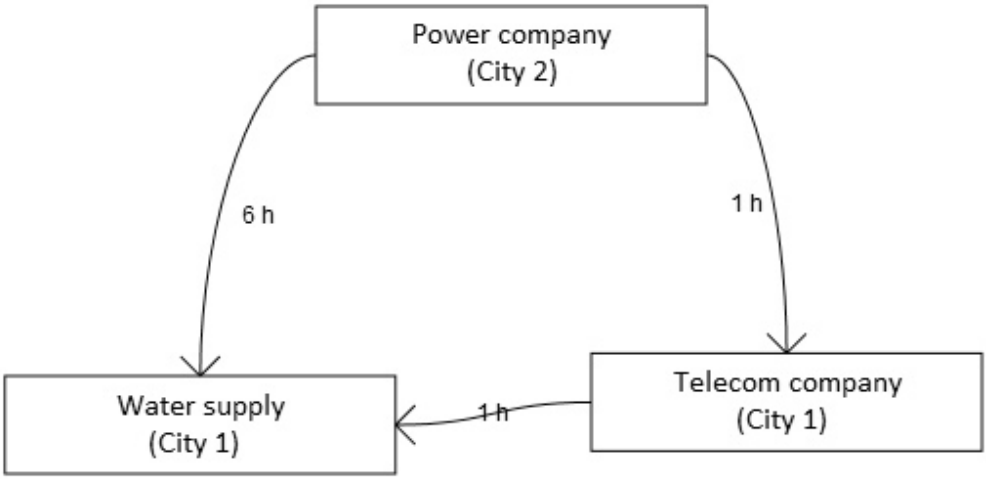
Käyttöliittymä on rakennettu valmiiden sovelluskehysten (software framework) päälle, mutta pohjimmiltaan ohjelmakoodi koostuu vain HTML-, CSS- ja JavaScript-kielistä. Toteutuksen visuaalinen ilme on toteutettu Bootstrapilla, joka on yksi suosituimmista käyttöliittymäsovelluskehyksistä [14]. Varsinainen toiminnallisuus on toteutettu JavaScript-kielen päälle rakennetulla AngularJS-ohjelmointikehyksellä [15]. Bootstrapin ja AngularJS:n lisäksi käyttöliittymä hyödyntää paljon vapaalla lähdekoodilla lisensoituja lisäosia, jotka mahdollistavat eri visualisointien esittämisen. Tilaympyrät, kartta, aikajana ja looginen kuvaaja ovat kaikki näiden lisäosien tuottamia visualisointeja.

Kuva 10: Aikanäkymän aikajana

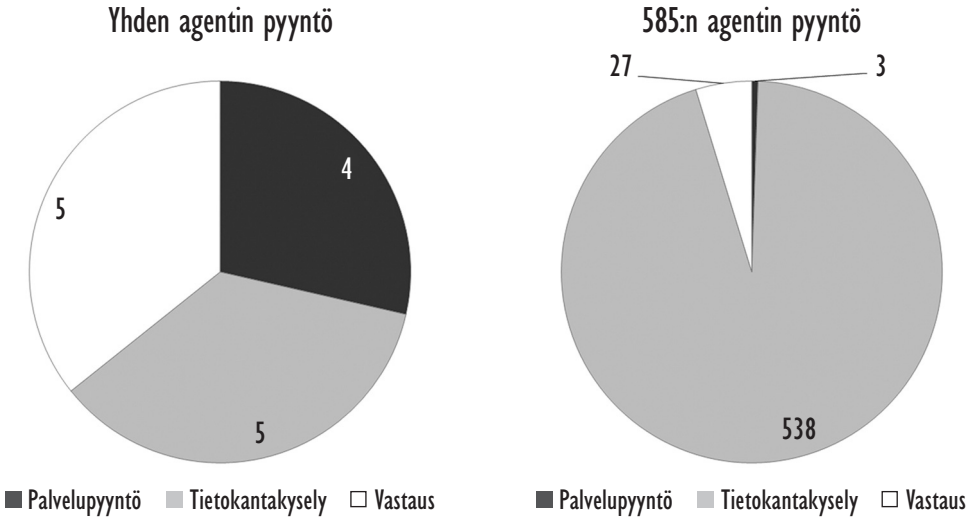
TESTAUS

Järjestelmän testaus toteutettiin suorituskykymittauksin sekä käyttäjätestein. Lisäksi eri järjestelmien integroitavuutta testattiin kehittämällä agentin lisäosat keräämään dataa yleisesti käytössä olevista teollisuuden ohjausjärjestelmistä (SCADA) ja tunkeutujanhavaitsemisjärjestelmistä (IDS). Sekä suorituskyky- että käyttäjätestien toteutusta ja tuloksia esitellään tarkemmin seuraavaksi.

Kuva 11: Karttanäkymä, jossa on näkyvillä ainoastaan vakavat tapahtumat



Kuva 12: Looginen näkymä kuvaa eri agenttien välisiä riippuvuuksia. Riippuvuuksiin liittyvät luvut kuvaavat aikaa, jonka järjestelmät pystyvät toimimaan riippuvuuden katketessa



Kuva 13: Viestinvälityksestä ja prosessoinnista aiheutuneet viiveet millisekunteina, kun rekisteröijältä pyydetään yhden tai useamman agentin tietoja

Suorituskykytestaus

Järjestelmän suorituskykytestit suoritettiin yhdellä virtuaalipalvelimella, jossa oli 2GB RAM muistia ja 8 suoritinydintä. Lähetetyt tapahtumat olivat yksinkertaisia ja pitivät sisällään tapahtuman luokittelun, vakavuuden, tapahtuman kuvauksen sekä tapahtumapaikan koordinaatit, jonka tuloksena yksittäisen tapahtuman koko oli noin yksi kilotavu.

Kapasiteettimittauksessa saaduissa tuloksissa yksi agentti pystyi tuottamaan keskimäärin 2477 viestiä sekunnissa, joista viestinvälittäjä (ks. Kuva 7) pystyi ohjaamaan eri komponenteille keskimäärin 665 tapahtumaa sekunnissa. Koska agentin tehtävä on tuottaa kohdejärjestelmästä ainoastaan valmiita tapahtumia raa'an datan sijasta, on lähes 2500 tapahtumaa/sekunti enemmän kuin riittävä sen viestinvälityskapasiteetiksi. Peruskäytössä voidaan olettaa, että agenteilta tulisi viestejä korkeintaan muutamia sekunnissa, joten viestinvälittäjän tarjoama 665 tapahtuman välityskykykin kykenee palvelemaan useita satoja agentteja. Testipalvelimen rajoitetut resurssit huomioiden on mitattu suorituskyky hyvä. Koska viestinvälittäjän toiminnallisuutta on mahdollista hajauttaa usealle palvelimelle ja saada siten lisää kapasiteettia, kykenee viestinvälittäjä tarjoamaan riittävän välityskapasiteetin koko järjestelmälle.

Viestinvälityskanavasta (ks. Kuva 5) aiheutuneet viiveet ovat olennaisessa osassa järjestelmän suorituskykyä arvioidessa. Mitatut viiveet pätevät myös muidenkin

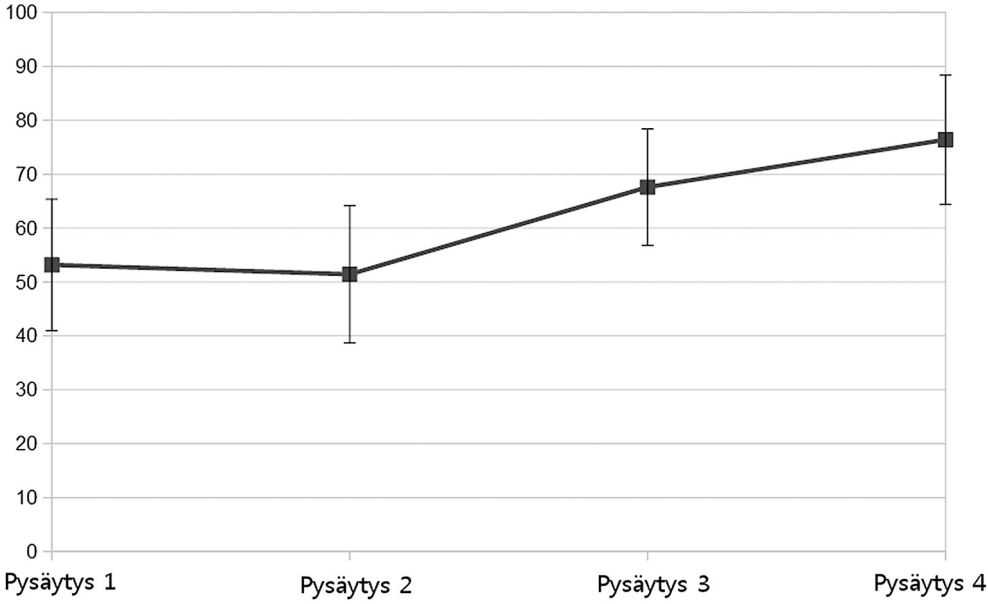
kuvan 5 komponenttien väliseen viestinvälitykseen, koska ne kaikki kommunikoivat samanlaisella mekanismilla. Kuvassa 13 on esitetty viiveitä pyydettäessä rekisteröijäkomponentilta tietoja ensin yhdestä ja sitten useasta agentista. Tuloksen arvot ovat millisekunteja ja niistä nähdään, että keskimääräinen palvelupyynnön tekeminen toiselle komponentille kesti 4 millisekuntia ja vastaus kyselyyn 5 millisekuntia. Rekisteröijän sisäisesti suorittama tietokantakysely (ks. Kuva 5), kesti tässä tapauksessa myös 5 millisekuntia. Pyydettäessä 585 agentin tietoja kerralla näkyy viiveissä huomattavat erot. Pyynnön viive on suuruusluokaltaan samanlainen myös toisessa testissä (3 ms), mutta tietokantakyselyn kesto on tällä kertaa huomattavasti suurempi isomman tietomäärän takia (538 ms). Lisäksi vastaus, joka pitää sisällään tiedot kaikista agenteista, aiheutti toisessa testissä huomattavasti suuremman viestinvälitysviiveen (27 ms) kuin palvelupyynnön välittäminen.

Testien perusteella viestinvälityskanava tarjoaa melko tehokkaan viestinvälityksen normaaleja palvelupyyntöjä suoritettaessa, jossa itse palvelupyyntö ja vastaus aiheuttavat alle 10 millisekunnin viiveen. Viiveet kuitenkin kasvavat luonnollisesti palvelupyynnön aiheuttamien taustaprosessien takia sekä suuremman datamäärän välittämisessä. Jopa melko suuret vastaukset kyetään välittämään suhteellisen tehokkaasti, koska 585 agentin tietojen välittäminen kestää ainoastaan 30 millisekuntia. On kuitenkin huomioitava, että suuret tietokantakyselyt aiheuttavat moninkertaista viivettä järjestelmään, joten niiden käyttöä tulee harkita.

Käyttäjätestaus

Järjestelmän käyttöliittymää arvioitiin ja testattiin kolmessa eri käyttäjätestaustapahtumassa. Käyttöliittymää kehitettiin iteratiivisesti, ja jokaisen iteraation jälkeen käyttöliittymän käytettävyyttä ja toiminnallisuuksia arvioitiin yhdessä käyttäjien kanssa. Ensimmäinen arviointi toteutettiin niin sanottuna kohderyhmähaastatteluna. Haastatteluun osallistui yksitoista sotatieteen kandidaattia, joilla oli kokemusta valvontaympäristöistä ja -käyttöliittymistä. Toisessa arvioinnissa kaksi kokeneempaa valvomohenkilöä kävivät käyttöliittymää läpi, ja arvioivat sen eri toimintoja ja niiden käytettävyyttä. Kolmannessa arvioinnissa käytettiin perinteisen käytettävyystestauksen lisäksi SAGAT-menetelmää [16], joka tuottaa numeerisia arvoja käyttäjän saavuttamasta tilannetietoisuudesta Endsleyn määrittämillä tasoilla, ks. Kuva 4. Toisessa ja kolmannessa arvioinnissa käytettiin myös SUS-kyselyä (system usability scale) [17], jolla mitataan käyttöliittymän käytettävyyttä kvantitatiivisesti.

Tilannetietoisuuden mittaamisessa käytetty SAGAT-menetelmä perustuu näyttöjen ajoittaiseen pysäyttämiseen, jonka jälkeen käyttäjältä kysytään kysymyksiä sen hetkiseen tilanteeseen liittyen. Koejärjestelyssä keskityttiin mittaamaan ainoastaan Endsleyn tilannetietoisuusmallin (Kuva 4) ensimmäistä ja toista tasoa, eivätkä tulokset sisällä kolmannen tason tilannetietoisuuden kehittymistä. SAGAT tulokset on esitetty kuvassa 14, joka kuvaa käyttäjän tilannetietoisuuden kasvua testitapahtuman edetessä. Kuvaajassa pys-



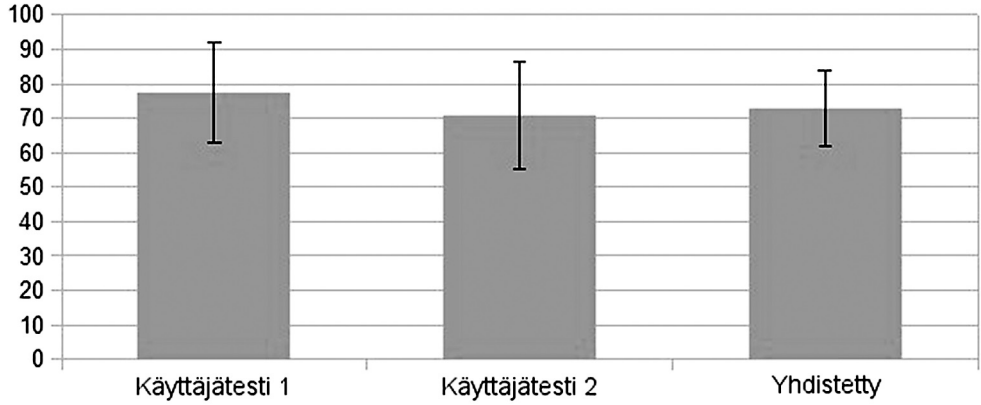
Kuva 14: Testitapahtumassa saadut SAGAT tulokset 95% luottamusväleineen. Y-akseli kuvaa oikeiden vastausten osuutta prosentteina kaikista kysymyksistä. X-akseli kuvaa testin etenemistä

tyakselilla on esitetty testihenkilöiden oikeiden vastausten osuutta prosenteissa kaikista kysymyksistä. Kuvan 14 nousevasta käyrästä nähdään, että SAGAT tulokset paranivat testitapahtuman edetessä, mikä voidaan tulkita tilannetietoisuuden paranemisena.

SUS-kyselyiden tulokset on esitetty kuvassa 15, jossa ensimmäisen kyselyn keskiarvotulos on 77,5 ja toisen kyselyn 71,3. Lisäksi kyselyiden yhdistetyksi tulokseksi saatiin 73,3. Tulokset eivät ole pronsettiosuuksia vaan kuvaavat käytettävyyttä lineaarisella asteikolla nollan ja sadan välillä. Tulokset sijoittuvat 7-portaisen asteikon kolmanneksi ylimmälle tasolle, joka vastaa kouluarvosanana hyvää. Tulos tarkoittaa, että käyttöliittymän käytettävyydessä kohdataan vähän tai ei lainkaan ongelmia. Lisäksi muutama testihenkilö kommentoi, että käyttöliittymä on yksinkertaisempi ja helpompi käyttää kuin Puolustusvoimissa yleisesti käytössä olevat järjestelmät.

JOHTOPÄÄTÖKSET

Lopullinen järjestelmä, joka tulisi käyttöön kansallisen kriittisen infrastruktuurin tilannekuvan luomiseksi, eroaisi todellisuudessa etenkin toteutusteknologioiltaan huomattavasti kehitetystä testijärjestelmästä. Kohdejärjestelmien agenttipohjainen integraatio yhdessä viestinvälittäjäarkkitehtuurin kanssa todet-



Kuva 15: SUS-tulokset kummassakin käyttäjäkyselyssä sekä niiden yhdistetty tulos 95 % luottamusväleineen

tiin kuitenkin testien avulla toimivaksi ratkaisuksi, joka sallii toteutuksen jopa kansallisessa laajuudessa. Agenttien kohdejärjestelmäkohtainen muokkaus ja riippuvuuksien kerääminen pääosin kohdejärjestelmäasiantuntijoiden toimesta on yksi tärkeimmistä tilannekuvajärjestelmän ominaisuuksista. Tällöin järjestelmäasiantuntijat saadaan osaksi tilannekuvan tuottamista, eikä tilannekuvajärjestelmän ylläpitäjien tarvitse koostua jokaisen järjestelmän asiantuntijoista.

Esitelty arkkitehtuuri ja sen toteutus toimivat hyvänä pohjana jatkokehitykselle, jossa paneudutaan tarkemmin riippuvuuksien mallintamiseen ja kokonaistilanteen muodostamiseen. Tämän lisäksi tutkimusta tulee jatkaa paremman tilannetietoisuuden aikaansaamiseksi mm. kokeellisten käyttöliittymien muodossa.

Lähdeluettelo

[1] Lewis, T. G. *Critical Infrastructure Protection in Homeland Security*: *defending:a networked nation.* John Wiley & Sons, 2006.

[2] Rinaldi, S. M., Peerenboom, J. P., ja Kelly, T. K. "*Indentifying, understanding, and analyzing critical infrastructure interdependencies. Control Systems," IEEE 21,* nro 6 (2001), pp. 11-25.

[3] Kosola, J. *Development of Technology and its effects to warfare 2015–2025.* Maanpuolustuskorkeakoulu, Sotatekniikan laitos, 2014, p. 57.

[4] Steinberg, A. N., Bowman, C.L., ja White, F. E. "*Revisions to the JDL Data Fusion Model,"* Proceedings of SPIE 3719 (1999), pp. 430-441.

[5] Giacobe, N. A. "Application of the JDL data fusion process model for cyber security," In: *SPIE Defense, Security, and Sensing.* International Society for Optics and Photonics, 2010, p. 77100R-77100R-10.

[6] Endsley, M. R. "Toward a theory of situation awareness in dynamic systems," *Human Factors: The Journal of the Human Factors and Ergonomics Society*, 1995, 37.1: 32-64.
[7] Langner, R. *To Kill a Centrifuge, A Technical Analysis of What Stuxnet's Creators Tried to Achieve*, The Langner Group, 2013, [viitattu 11.9.2014], Saatavissa: http://www.langner.com/en/wp-content/uploads/2013/11/To-kill-a-centrifuge.pdf.
[8] Luckham, D. C., ja Frasca, B. *Complex event processing in distributed systems*, Computer Systems Laboratory Technical Report CSL-TR-98-754, Stanford University, Stanford 28 (1998).
[9] Spring Framework, Pivotal Software, 2014, [viitattu 11.9.2014], Saatavissa: http://spring.io/.
[10] Apache ActiveMQ, Apache Software Foundation, [viitattu 11.9.2014], Saatavissa: http://activemq.apache.org/.
[11] Apache Camel, Apache Software Foundation, [viitattu 11.9.2014], Saatavissa: http://camel.apache.org/.
[12] Hibernate ORM, Hibernate, [viitattu 11.9.2014], Saatavissa: http://hibernate.org/orm/.
[13] Endsley, M. R., "Situatio awareness-oriented design," The Oxford Handbook of Cognitive Engineering, 2013, p. 272.
[14] R. M. Lerner, "At the forge: Twitter bootstrap," Linux journal, nro 6, p. 218, 2012.
[15] AngularJS, Google, [viitattu 11.9.2014], Saatavilla: https://angularjs.org/.
[16] Endsley, M. R., "Situation awareness global assessment technique (SAGAT)," *Aerospace and Electronics Conference, 1988. NAECON 1988, Proceedings of the IEEE 1988 National*, 789–795, IEEE (1988).
[17] Brooke, J., "SUS—a quick and dirty usability scale," *Usability Evaluation in Industry* 189, 194 (1996).