TEKNILLINEN KORKEAKOULU
Sähkötekniikan osasto

Jarmo Suihkonen

SYNKRONISTA DIGITAALISTA HIERARKIAA PALVELEVAN SOLMUN TIEDONVARMISTUS

Diplomityö, joka on jätetty opinnäytteenä tarkastettavaksi diplomi–insinöörin tutkintoa varten Espoossa

Työn valvoja — Seppo J. Halme

Työn ohjaaja — Tuomo Kaukonen

HELSINKI UNIVERSITY OF TECHNOLOGY

ABSTRACT OF THE MASTER'S THESIS

Author: Jarmo Suihkonen

Name of the thesis:

Parameter back–up management of a node serving synchronous digital hierarchy

Date: September 1st, 1993 Number of pages: 77

Faculty: Electrical Engineering

Professorship: Communications Engineering

Supervisor: professor Seppo J. Halme

Instructor: M.A. Tuomo Kaukonen

The meaning of this master's thesis has been to design a C++ –based parameter backup management software, which provides the software of a SDH node possibility to store their essential information in non–volatile memory, thus improving fault tolerance of the system.

The node environment in question is an embedded system put together from several plug–in units containing a MC68302 microprocessor and a real–time operating system. The backup management service allows a limited replacement of any plug–in unit with another without the original information being lost or changed in the node.

The backup management software is duplicated into all plug–in units, and can communicate with each other using an Ethernet connection built in the rack backplane. Attention has been paid to the inter–unit operations and their fault tolerance.

Keywords: backup, C++, EEPROM, embedded, MC68302, object, real–time, SDH

TEKNILLINEN KORKEAKOULU DIPLOMITYÖN TIIVISTELMÄ

Tekijä: Jarmo Suihkonen

Työn nimi: Synkronista digitaalista hierarkiaa palvelevan solmun tiedonvarmistus

Päivämäärä: 1.9.1993 Sivumäärä: 77

Osasto: Sähkötekniikan osasto

Professuuri: Tietoliikennetekniikka

Työn valvoja: professori Seppo J. Halme

Työn ohjaaja: FK Tuomo Kaukonen

Diplomityön tarkoituksena on ollut suunnitella erääseen sulautettuna järjestelmänä toteutettuun synkronista digitaalista hierarkiaa palvelevaan solmuun C++ −kielinen MC68302−mikroprosessorilla suoritettava tiedonvarmistusohjelmisto, joka tarjoaa reaaliaikakäyttöjärjestelmäympäristössä toimiville ohjelmistokomponenteille mahdollisuuden tallettaa solmun toiminnan ja verkonhallinnan kannalta keskeisiä tietoja haihtumattomaan EEPROM−muistiin.

Solmu koostuu vaihtelevasta yhdistelmästä keskenään kommunikoivia pistoyksiköitä, joille tiedonvarmistusjärjestelmä on monistettu. Ohjelmistossa on kiinnitetty huomiota pistoyksiköiden väliseen toimintaan ja sen vikasietoisuuteen.

Tiedonvarmistuspalvelu mahdollistaa minkä tahansa pistoyksikön rajoitetun vaihtamisen toiseen ilman, että sille talletetut tiedot katoavat solmusta tai muuttuvat pistoyksikön mukana.

Avainsanat: C++, EEPROM, MC68302, olio, reaaliaika, SDH, sulautettu, tiedonvarmistus

Alkulause

Tämä diplomityö tehtiin Nokia Telecommunications Oy:n siirtojärjestelmien tuotekehitysosastolla. Haluan kiittää esimiestäni FK Tuomo Kaukosta projektin aikana saamastani kaikenpuolisesta tuesta.

Haluan kiittää myös professori Seppo J. Halmetta työn eri vaiheissa käymistämme keskusteluista.

Kollegoilleni olen kiitollinen kommenteista, jotka vaikuttivat työn kulkuun merkittävästi. Erityisen paljon oli apua Jaripekka Turtiaisen antamasta reaaliaikajärjestelmän mallintamista koskevasta tuesta.

Espoossa 1.9.1993

Jarmo Suihkonen

Sisällysluettelo

SYMBOLI- JA LYHENNELUETTELO

2M	Fyysinen liitäntäpistoyksikkö 2 Mbit/s kanaville (*2 Mbit/s Interface Unit*).
2MTA	Fyysinen päätesovitustoiminteita sisältävä liitäntäpistoyksikkö 2 Mbit/s kanaville (*2 Mbit/s Interface Unit with Terminal Adapter*).
ADM	Syöttöpudotuslaite (*Add–Drop Multiplexer*).
ANSI	*American National Standards Institute*
API	Sovellusrajapinta (*Application Program Interface*).
APS	Automaattinen siirtoyhteyden varmennus (*Automatic Protection Switching*).
ASW	Sovellusohjelmisto (*Application Software*).
BaM	Solmun tiedonvarmistuspalvelut tuottava ohjelmistokomponentti (*Back-up Manager*).
CBM	Eräs SDH–solmun ohjelmistokomponentti (*Control Block Manager*).
CCITT	*Comité Consultatif International de Télégraphique et Téléphonique*
CU	Solmun ohjainpistoyksikkö (*Control Unit*).
DARTS	*Design Approach for Real–Time Systems*
DCC	*Data Communication Channel*
DCC_M	*Multiplexer Data Communication Channel*
DCC_R	*Regenerator Data Communication Channel*
DXC	Numeerinen ristikytkentä (*Digital Cross–connect*).
ECC	Sulautettu ohjauskanava (*Embedded Control Channel*).
EEPROM	*Electrically Erasable Read–Only Memory*
HLTD	Tiedonvarmistusjärjestelmän korkean tason päätöksentekoelin (*High–level Topology Decider*).
IUS	Pistoyksikön solmutason tiedonvarmistusjärjestelmäpalvelin (*Inter–unit Server*).
IUCM	Solmun sisäistä pistoyksiköiden välistä sanomanvaihtopalvelua tarjoava ohjelmistokomponentti (*Inter–unit Communication Manager*).
LAN	Lähiverkko (*Local Area Network*).
LS	Pistoyksikön paikallinen tiedonvarmistuspalvelin (*Local Server*).
MC68302	Motorolan valmistama mikroprosessori.
MeM	EEPROM–ajurina toimiva ohjelmistokomponentti (*Memory Manager*).
MIT	Hallintatietopuu (*Management Information Tree*).
MSOH	Kanavointilaitteiden välinen siirto–otsikko (*Multiplexer Section Overhead*).
MUX	Kanavointilaite (*Multiplexer*).
NFL	Solmutoimintakerros (*Node Functionality Layer*).
OCM	Solmun sisäistä oliokonfiguraatiota seuraava ohjelmistokomponentti (*Object Configuration Manager*).

OS68	Enea Datan Motorola MC68000–sarjan mikroprosessoreita tukeva reaaliaikakäyttöjärjestelmä.
REG	Toistin (*Regenerator*).
RSOH	Toistimelta lähtevä siirto–otsikko (*Regenerator Section Overhead*).
SDH	Synkroninen digitaalinen hierarkia (*Synchronous Digital Hierarchy*).
SMN	SDH–hallintaverkko (*SDH Management Network*).
SOH	STM–kehyksen siirto–otsikko (*Section Overhead*).
SONET	Pohjois–Amerikassa käytettävä optinen synkroninen siirtojärjestelmä (*Synchronous Optical Network*).
SSW	Fyysinen ristikytkentäpistoyksikkö (*System Switch Unit*).
STM	Synkroninen siirtomoduli (*Synchronous Transport Module*).
STM–1	1) SDH:n perussiirtokehys 2) Fyysinen liitäntäpistoyksikkö 155,520 Mbit/s optiselle signaalille (*STM–1 Optical Interface Unit*).
STM–1E/140M	Fyysinen liitäntäpistoyksikkö sähköiselle 155,520 Mbit/s tai 139,368 Mbit/s signaalille (*STM–1/Electrical Interface Unit*).
STM–4	Fyysinen liitäntäpistoyksikkö 622,080 Mbit/s optiselle signaalille (*STM–4 Optical Interface Unit*).
SU	Valinnainen palvelupistoyksikkö (*Service Unit*).
TM	Päätekanavointilaite (*Terminal Multiplexer*).
TMN	Televerkon hallintaverkko (*Telecommunications Management Network*).
TSW	Fyysinen ristikytkentäpistoyksikkö (*Cross–connect Unit*).

1 JOHDANTO

Yhdysvalloissa alettiin 1980–luvulla kehittää optisen liitännän standardia synkronisia siirtoverkkoja varten. Työn tuloksena syntyi ANSI:n SONET–standardi (*Synchronous Optical Network*), joka määrittelee optisen liitännän lisäksi verkonhallinnan ja perinteisten plesiokronisten signaalien liittämisen synkroniseen verkkoon. CCITT julkaisi vuonna 1988 SONET:iin perustuvat suositukset synkronisesta digitaalisesta hierarkiasta (SDH), joka leviää Pohjois–Amerikan ulkopuolella. SDH on yhteensopiva ANSI:n SONET:n kanssa, joten standardit mahdollistavat ensimmäisen maailmanlaajuisesti yhteensopivan suurkapasiteettisen siirtojärjestelmäverkoston luomisen.

Toisin kuin plesiokronisissa järjestelmissä, verkonhallinnalla on synkronisen digitaalisen hierarkian verkossa suuri merkitys. Teleoperaattoreiden kannalta keskitetty verkonhallinta on taloudellinen ratkaisu ja houkuttelee uuden tekniikan käyttöönottoon.

Verkonhallintajärjestelmä edellyttää siirtoverkon solmuilta älykkyyttä, jonka osana on solmujen kyky muistaa nille annetut tehtävät ja asetukset. Tämän diplomityön tarkoituksena on ollut suunnitella erääseen SDH–järjestelmään solmun sisäinen ohjelmallinen tiedonvarmistusjärjestelmä, joka toteuttaa nämä vaatimukset tarjoamalla solmun muille ohjelmistokomponenteille tiedostopohjaisen muistipalvelun.

Selostuksen aluksi on esitelty SDH–järjestelmän yleisiä piirteitä, minkä jälkeen määritellään tiedonvarmistusjärjestelmälle asetettavat vaatimukset. Tältä pohjalta on lähdetty pohtimaan mahdollisuuksia käytännön ratkaisun tueksi ja lopuksi esitellään suunnittelun tuloksena syntynyt käytännön ratkaisu.

2 SDH–JÄRJESTELMÄ JA TIEDONVARMISTUSTARPEET

2.1 Verkko ja sen hallinta

SDH:n peruspiirteet on määritelty CCITT:n standardeissa: siirtonopeudet standardissa G.707, verkko–solmu–rajapinta standardissa G.708 ja kanavointi standardissa G.709 /1//2//3/.

Tiedonsiirto SDH:ssa perustuu synkronisen siirtokehyksen (STM, *Synchronous Transport Module*) käyttöön. Siirrettävä data kanavoidaan STM–peruskehykseen (STM–1), joka lähetetään optisesti eteenpäin; kehyksessä siirtyy samanaikaisesti myös verkonhallinta– ja valvontatietoa. Yhteen STM–1–kehykseen multipleksoidaan 63 plesiokronista nopeuden 2 Mbit/s signaalia.

SDH–järjestelmän korkeampien hierarkiatasojen siirtonopeudet ovat perusnopeuden kokonaislukumonikertoja, joten kanavointi hierarkiatasolta toiselle ei edellytä täytebittien käyttöä. Käytännön kaupallisissa tuotteissa tarjotaan alkuvaiheessa perusjärjestelmän lisäksi monikertoja neljä ja kuusitoista (taulukko 1).

Kanavointi esimerkiksi alimmalta tasolta tasolle 16 voidaan tehdä tason neljä kautta; kanavoinnin periaatteet on kuitenkin määritelty siten, että samaan lopputulokseen päädytään myös kanavoimalla perustason signaalit suoraan tasolle 16.

Taulukko 1 Tällä hetkellä kaupallisesti tarjottavia SDH:n hierarkiatasoja vastaavat siirtonopeudet /1/

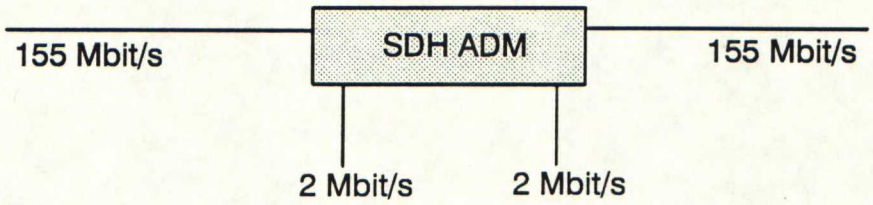
Hierarkiataso	Siirtonopeus Mbit/s
STM–1	155,520
STM–4	622,080
STM–16	2488,320

SDH–verkon solmut voidaan jakaa kahteen päätyyppiin:

- kanavointilaitteisiin (MUX), joita ovat päätekanavointilaitteet (TM), syöttöpudotuslaitteet (ADM) ja ristikytkentälaitteet (DXC), sekä

- toistimiin (REG).

Kanavointilaitteet hoitavat sekä alivirtojen kanavoinnin STM–kehyksiin että niiden purkamisen. SDH:n perusideologiaan kuuluu, että syöttöpudotuslaitteen avulla päästään kuvan 1 mukaisesti käsiksi esimerkiksi mihin tahansa nopeuden 2 Mbit/s alivirtaan tarvitsematta purkaa koko STM–kanavointia, mikä on merkittävä etu plesiokroniseen järjestelmään nähden.



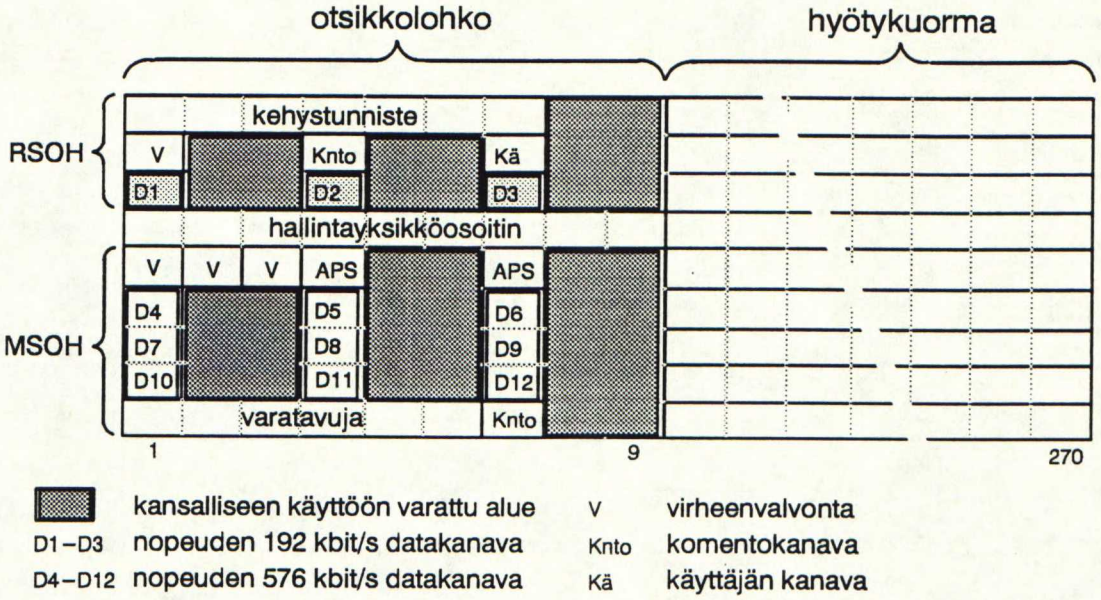
Kuva 1 *Yksittäisen alivirran erottaminen synkronisen digitaalisen hierarkian syöttöpudotuslaiteen avulla.*

Ristikytkentälaitteen avulla esimerkiksi STM–1 –alivirrat voidaan kytkeä mielivaltaisesti. Ristikytkentää ohjataan hallintaverkon kautta; käytännön kytkennät lienevät useinmiten puolikiinteitä ja niitä muutetaan esimerkiksi verkon jonkun osan kuormittuessa selvästi muita enemmän tai muuttuessa käyttökelvottomaksi, sekä yhteyksiä muutettaessa.

SDH–järjestelmän verkonhallinnassa osa hallintasovelluksesta ja pääosa konfiguraatioon liittyvästä tiedosta – muun muassa ristikytkentätilanne, hälytykset, siirtolaitteiden kalibrointiparametrit ja elementtien tunnisteet – on hajautettu solmuihin, joten solmuilla täytyy olla kyky muistaa keskeiset asetuksensa. Nopean uudelleenkäynnistymisen saavuttamiseksi esimerkiksi huoltotyön tai jännitekatkoksen jälkeen kunkin solmun on joko muistettava tai kyettävä pikaisesti selvittämään myös naapurisolmujen olemassaolo ja suhteellinen sijainti.

SDH–järjestelmän STM–1–kehykset (kuva 2) tarjoavat otsikkokentissään (SOH, *Section Overhead*) melko suuren verkon valvontaan ja hallintaan varatun siirtokapasiteetin, 81 tavua koko kehyksen 2430 tavusta /2/. CCITT:n standardin mukaisesti siirrossa valvotaan sekä siirtolinkin kuntoa että hyötykuorman läpimenoa. Siirto–otsikon sisällä kulkee:

- virhevalvontatietoa,
- kaksi 64 kbit/s huoltopuhelinyhteyttä (komentokanavat),
- automaattisen siirtoyhteyden varmennuksen (APS) merkinantotietoa,
- vapaavalintaista käyttäjän dataa ja
- verkonhallinnan hyödyntämä DCC–kanava.



Kuva 2 STM–1–tason siirtokehyksen otsikkorakenne

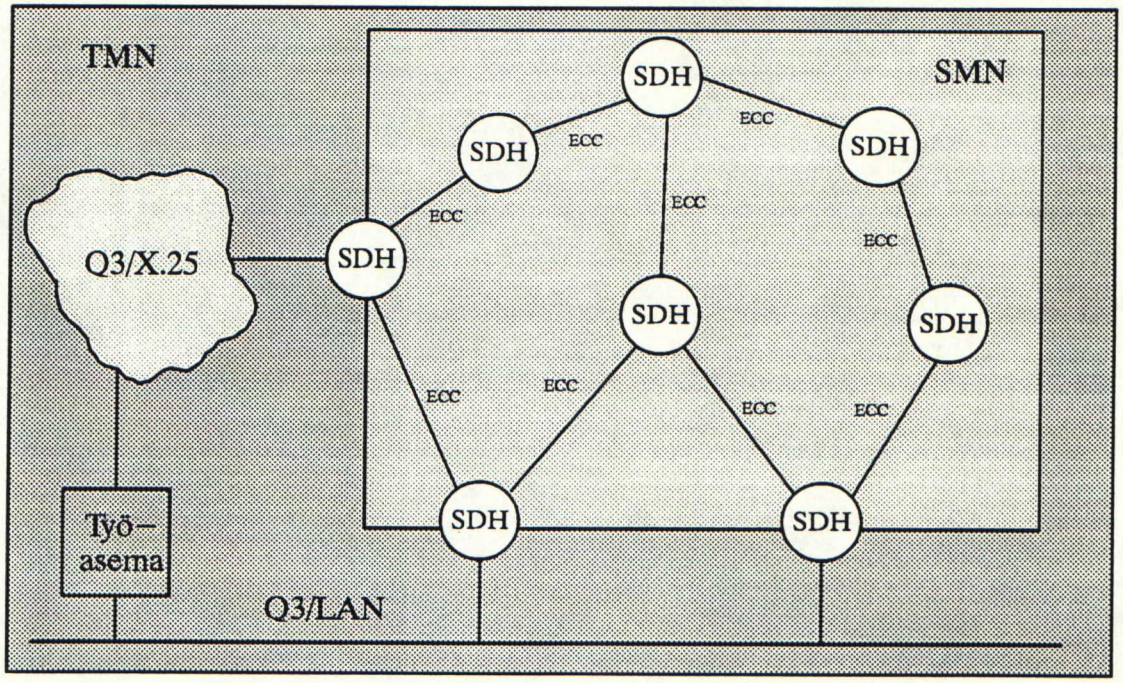
Vastakkaisten kanavointilaitteiden välillä kulkee kanavointilinkkiotsikoita (MSOH), joiden avulla valvotaan ja hallitaan päästä päähän yhteyttä (kuva 3). MSOH:n sisältämät yhdeksän DCC_M–kanavatavua (tavut D4–D12 kuvassa 2) tarjoavat nopeuden 576 kbit/s yhteyden, jota hyödynnetään verkonhallintayhteyksissä (ECC, *Embedded Control Channel*).

Kuva 3 Päästä päähän kulkeva kanavointilinkki ja toistinyhteyden valvontalinkit.

Toistimelta naapurisolmuille kulkee toistinlinkkiotsikoita (RSOH), joita käyttäen valvotaan välilinkkejä. RSOH sisältää vain kolme DCC_R–kanavatavua (tavut D1–D3 kuvassa 2) tarjoten nopeuden 192 kbit/s yhteyden, mutta se on DCC_M:stä poiketen kaikkien verkon elementtien käytettävissä.

Plesiokronisten järjestelmien keskitettyyn hallintaan ei ole olemassa vakiintunutta laitevalmistajariippumatonta ratkaisua, joten SDH–järjestelmä, jossa kaikkien valmistajien valmistamia solmuja voidaan käyttää verkon osina ja ohjata yhdestä tai useammasta paikasta, on tässä suhteessa uranuurtaja. Tulevaisuudessa mahdollisesti saadaan aikaan standardisoitu menettely, joka mahdollistaa kaikkien siirtojärjestelmien yhdistämisen saman verkonhallinnan alaisuuteen (TMN, *Telecommunications Management Network*). TMN:n osana toimivaa SDH–laitteiden ja niiden välisten hallintakanavien muodostamaa kokonaisuutta kutsutaan SDH–hallintaverkoksi (SMN, *SDH Management Network*).

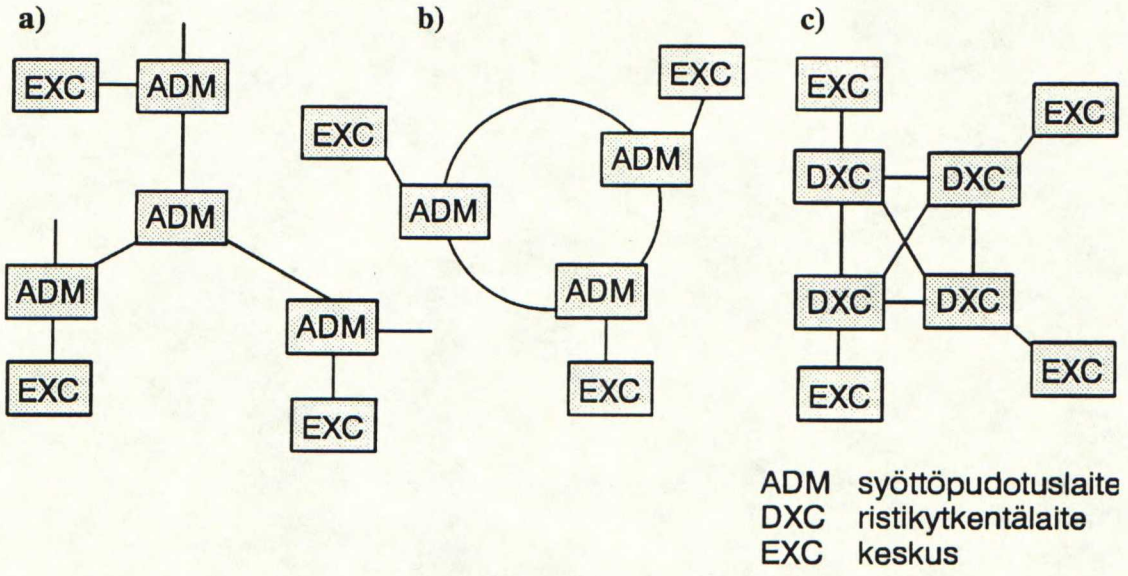
Verkkoa hallitaan joko paikallisesti tai verkon yli käyttäen hallintatyöasemaa, joka kytkeytyy SDH–hallintaverkkoon joko lähiverkko– tai X.25–yhteyden kautta Q3–protokollapinon yli (kuva 4). Verkossa hallintatieto kulkee ECC–väyliä pitkin.



Kuva 4 SDH–hallintaverkon periaatteellinen rakenne /4/.

SDH–teknologialla on mahdollista toteuttaa monipuolisia verkkoratkaisuja, mutta alkuvaiheessa järjestelmiä käytetään pääasiassa suurkapasiteettisina runkolinjoina, joita hyödyntävät mm. teleoperaattorit, datasiirtopalveluiden suurkuluttajat ja kaapelitelevisioyhtiöt.

Useimmilla teleoperaattoreilla on käytettävissään runsaasti vanhoja siirtojärjestelmiä SDH–verkon varmistukseen. Tällöin SDH–verkon topologia on pääsääntöisesti aluksi yksinkertainen – esimerkiksi väylä tai kuvan 5a mukainen tähti – mutta todennäköisesti laajenee siten, että myös mahdolliset varmistavat yhteydet hoidetaan SDH:n avulla, jolloin muun muassa päästään hyödyntämään hallintajärjestelmää reitityksen ohjauksessa. Jos verkossa on solmuja vähän, jopa täydellinen kytkentä on mahdollista toteuttaa ristikytkentälaitteita käyttäen (kuva 5c).



Kuva 5 *Esimerkkejä erilaisista verkkotopologioista: a) tähti, b) rengas ja c) täydellinen kytkentä.*

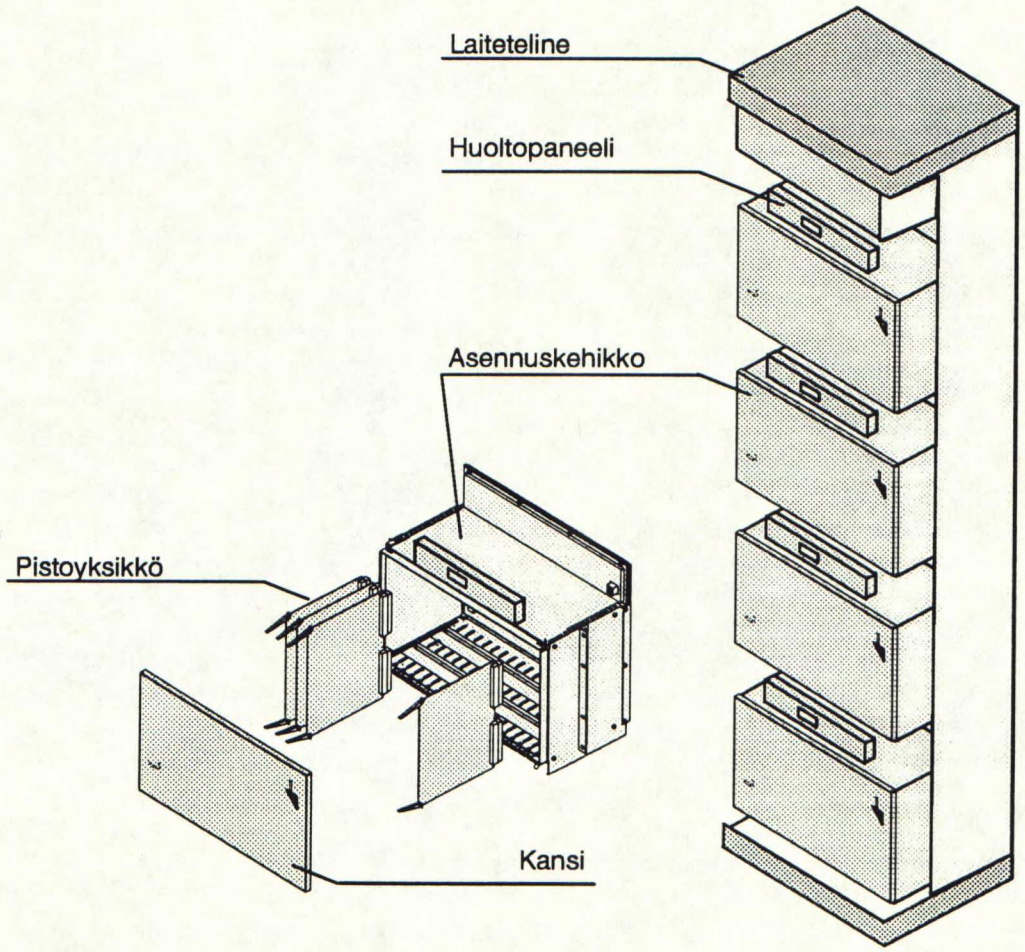
Mutkikkaat verkkorakenteet asettavat verkonhallintaan liittyvälle ohjelmistolle kovia vaatimuksia ja kriittiseksi tekijäksi voi nousta mm. vasteaika, johon vaikuttaa myös solmujen tiedonvarmistusjärjestelmän toiminta.

2.2 Fyysinen solmu

Nokia Telecommunicationsin SDH–solmu, johon liittyen tämä diplomityö on tehty, kalustetaan pistoyksiköitä käyttäen yhteen asennuskehikkoon (kuva 6). Pistoyksikkövalinnasta riippuen solmu toimii joko toistimena tai halutunlaisena kanavointilaitteena /6/.

Kullakin pistoyksiköllä sijaitsee oma Motorola MC68302–mikroprosessori /7/, joka suorittaa yksikkökohtaista koodia. Asennuskehikon takalevyssä oleva Ethernet–verkko kytkee pistoyksiköt yhteen mahdollistaen yksiköiden yhteistoiminnan.

Kaikki pistoyksiköt sisältävät EEPROM–muistia ja tarvitsevat palvelua, joka mahdollistaa tietojen tallentamisen haihtumattomaan muistiin ja palauttamisen sieltä.



Kuva 6 Nokia Telecommunicationsin SDH–laite ('Synfonet')

Käytettävät pistoyksikkötyypit ovat:

- Ohjausyksikkö (CU, *Control Unit*), joka sisältää muun muassa Q3–protokollapinon ja sovellusohjelmistot, suojaus– ja synkronointitoiminteet; voidaan kahdentaa /5/.
- Tilatason ristikytkentäyksikkö (SSW, *System Switch Unit*), joka voidaan kahdentaa /6/.
- Liitäntäyksikkö STM–1 –tason optiselle signaalille (STM–1, *STM–1 Optical Interface Unit*) /6/.
- Liitäntäyksikkö sähköiselle STM–1 –tason tai nopeuden 139,368 Mbit/s signaalille (STM–1E/140M, *STM–1/Electrical Interface Unit*) /6/.
- Liitäntäyksikkö STM–4 –tason optiselle signaalille (STM–4, *STM–4 Optical Interface Unit*) /6/.
- Valinnainen palveluyksikkö (SU, *Service Unit*), joka sisältää muun muassa huoltopuhelin– ja datakanavaliitännät /8/.
- Aikatason ristikytkentäyksikkö (TSW, *Time Switch Unit*) /6/.
- Liitäntäyksikkö kuudelletoista nopeuden 2 Mbit/s kanavalle (2M, *2 Mbit/s Interface Unit*) /6/.
- Päätesovitustoiminteita sisältävä liitäntäyksikkö kuudelletoista nopeuden 2 Mbit/s kanavalle (2MTA, *2 Mbit/s Interface Unit with Terminal Adapter*) /6/.

STM–1–päätekanavointilaitetta tarvittaessa asennuskehikko kalustetaan CU:lla, optisella tai sähköisellä STM–1–tason liitäntäyksiköllä, yhdellä 2MTA–pistoyksiköllä ja tarpeen mukaan 1–3 tavallisella 2M–yksiköllä /6/. Valinnaisena lisättävä SU antaa mm. mahdollisuuden huoltopuhelinyhteyteen ja erillinen jännitteensyöttöpistoyksikkö mahdollistaa asennuskehikon käytön ilman laitetelinettä.

STM–4–päätekanavointilaitteessa STM–1–pistoyksikkö korvataan STM–4 –yksiköllä ja 2M–yksiköt pääsääntöisesti 1–4 kappaleella STM–1– ja STM–1E/140M–yksiköitä /6/. STM–1–tason korttien tilalla ja rinnalla on mahdollista käyttää myös 2M– tai 2MTA–yksiköiden yhdistelmiä, jolloin mahdollisesti vajaassa käytössä olevaan päätekanavointilaitteeseen voidaan suoraan liittää nopeuden 2 Mbit/s signaaleja.

STM–1 –tason toistinlaitteessa tarvitaan CU:n ja mahdollisen SU:n lisäksi kaksi optisen signaalin STM–1–yksikköä, jotka STM–4–toistimessa korvataan suurempikapasiteettisilla STM–4–yksiköillä /6/.

Syöttöpudotuslaitteeksi kalustettavaan asennuskehikkoon sijoitettavien pistoyksiköiden tyypit ja lukumäärät määräytyvät sen mukaan, minkä tason signaaleja halutaan käsitellä; peruskokoonpanoon kuuluu kuitenkin kahdennettu CU ja vähintään yksi SSW–yksikkö. Näiden ja mahdollisen SU:n lisäksi solmuun voi kuulua erilaisina yhdistelminä 1–4 kappaletta STM–4–yksiköitä, 1–15 kappaletta optisen signaalin STM–1–yksiköitä, 1–8 kappaletta STM–1E/140M–yksiköitä, 1–15 kappaletta TSW–yksiköitä ja 1–8 kappaletta 2M– ja 2MTA–yksiköitä /6/.

Solmu voidaan kalustaa myös nopeuden 64 kbit/s ristikytkentälaitteeksi, jolloin tarvitaan CU:n lisäksi yksi TSW–yksikkö, yksi 2MTA–yksikkö ja enintään kolme 2M–yksikköä /6/. Koska jokainen 2M– ja 2MTA–liitäntäyksikkö sisältää 16 kappaletta nopeuden 2 Mbit/s liitäntöjä, solmu kykenee 63 nopeuden 2 Mbit/s kanavan sisältämien nopeuden 64 kbit/s kanavien ristiin kytkemiseen.

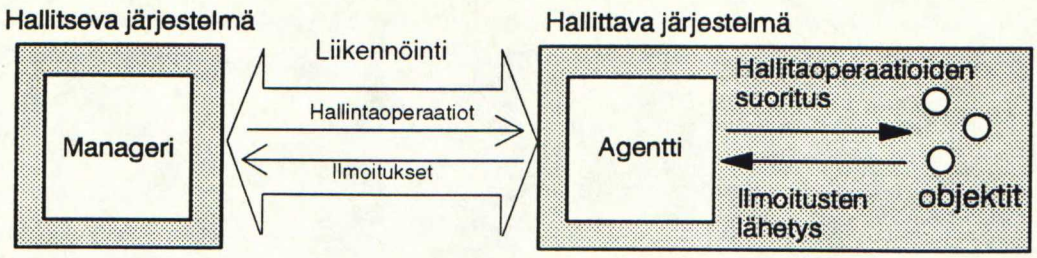
2.3 Solmun ohjelmisto

Suuri osa varhaisemmissa järjestelmissä laitteistollisesti toteutetuista toiminteista hoidetaan ohjelmallisesti, joten tulkittaessa SDH–solmun ulkoista käyttäytymistä rajaten transmissio pääosin tarkastelun ulkopuolelle, siirrytään laitteistojen sisältämän ohjelmiston alueelle. Ohjelmistonkehityksen nousuun ovat johtaneet tehokkaat mikroprosessorit, halvat muistipiirit, kehitystyökalujen monipuolistuminen ja ohjelmistotototeutuksen joustavuus – laitteen toiminnallisuutta voidaan muuttaa ja mahdollisia virheitä korjata vaihtamalla muistipiirit uuden ohjelmiston sisältäviksi tai lataamalla uusi koodi vanhoille muistipiireille.

SDH sisältää loogisesti ja ohjelmistotasolla monia osia, joissa jokin järjestelmä tai olio hallitsee toista. Tätä kaikilla tasoilla yleistä suhdetta kutsutaan agentti–manageri –suhteeksi (kuva 7) /9/. Jokainen SDH–verkon solmu on verkonhallintajärjestelmän agentti.

Agentti–manageri –konsepti kuvaa hallitsevan järjestelmän (manageri) ja hallittavan järjestelmän (agentti) yhteistoimintaa ja vuorovaikutusta tietyn toiminnan toteuttamiseksi. Ajatuksena on, että manageri antaa agentille tehtäviä hoidettavaksi ja agentti lähettää managerille ilmoituksia tehtävän etenemisestä ja tuloksista. Käytännössä agentilla on managerin rooli tiettyihin olioihin nähden, jotka hoitavat osia agentin saamasta tehtävästä. Tällä tavalla tehtävän suoritus eriytyy ja yleinen hallintaoperaatio tarkentuu sarjaksi pieniä tehtäviä. Ohjelmistoissa tämä ajattelutapa so-

veltuu erityisen hyvin käytettäväksi sulautetuissa oliosuuntautuneesti toteutettavissa järjestelmissä.



Kuva 7 Agentti–manageri –konsepti

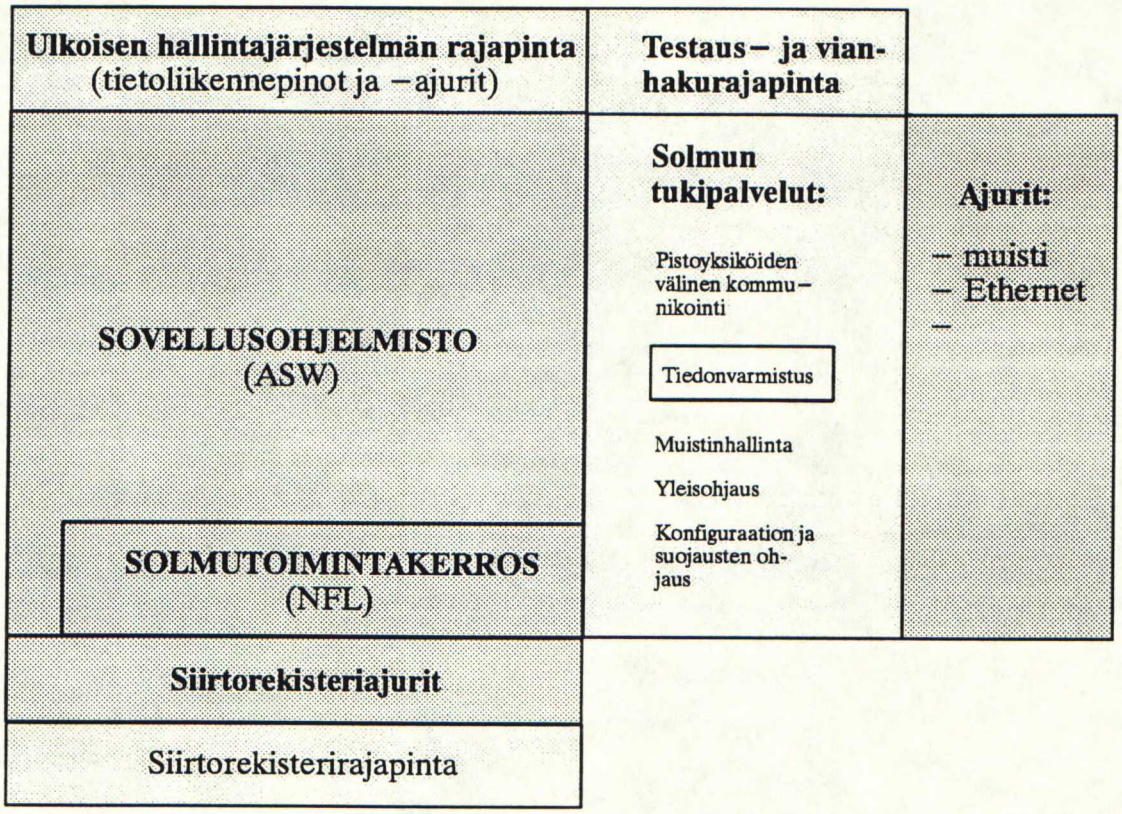
Solmun ja verkon välisen standardoidun rajapinnan takana eri valmistajien tekemät solmut poikkeavat toisistaan; yhteensopivuus verkonhallintaan päin taataan standardin mukaisella tietoliikennepinolla, johon voidaan kohdistaa vakioituja operaatioita eli verkonhallinta näkee kunkin solmun resursseineen hallittuna oliona (*Managed Object*).

Nokian ratkaisussa (kuva 8) hallintarajapinnan toiselta puolelta löytyy sovellusohjelmisto, joka muodostaa solmun toiminnallisen rungon vastaten hallintaverkon operaatioihin. Käytännössä sovellusohjelmisto luo solmun rajapintaan kulissin, jossa kaikki solmun resurssit verkonhallintaan päin näkyvät hallittuina nimettyinä olioina muodostaen hallintatietopuun (MIT, *Management Information Tree*) /10/ ja sovellusohjelmiston tehtäviin kuuluu solmun toiminnan seuraaminen ja käskytys.

Kukin solmu sisältää tietyt hallitut oliot ja sovellusohjelmisto vastaa siitä, että verkonhallinta tavoittaa solmun resurssit. Siksi sovellusohjelmisto pyrkii mahdollisuuksien mukaan peittämään muun muassa satunnaiset ei–fataalit käyttökatkot (esimerkiksi vioittuneen pistoyksikön korvaaminen toisella); ne eivät aiheuta koko solmun muuttumista hallitsemattomaksi.

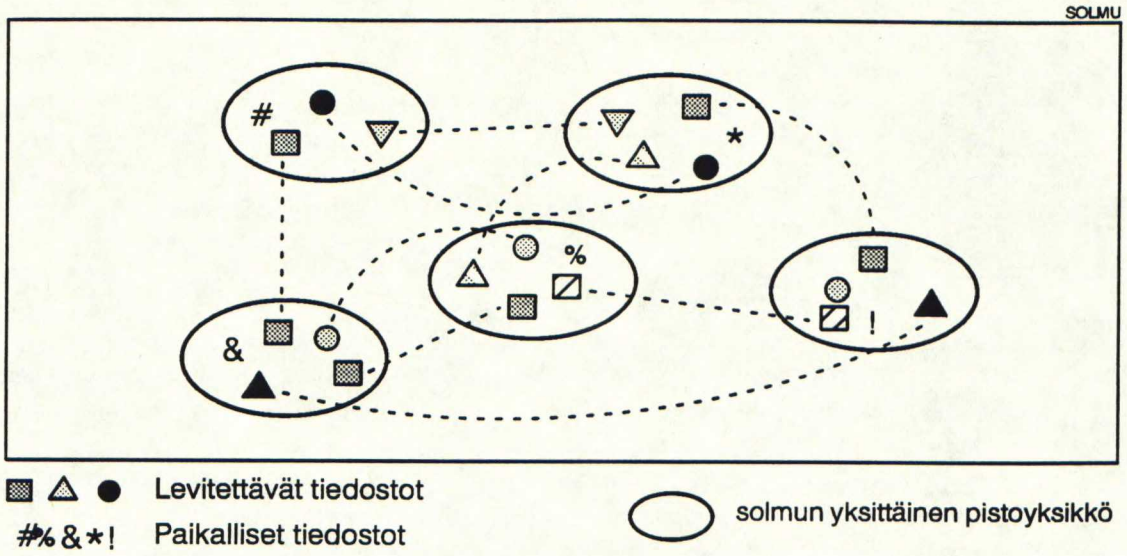
Käytännössä sovellusohjelmisto sisältää useita agentti–manageri –mallin mukaisia olioita, jotka vastaavat solmun todellisia resursseja /11/. Nämä oliot on hajautettu solmun eri pistoyksiköille siten kuin niiden tehokas toiminta edellyttää. Solmun muistiominaisuudet, joihin edellä viitattiin, ovat pääosin sovellusohjelmiston vastuulla; hallitut oliot tallentavat muuttuneet asetuksensa tiedonvarmistusjärjestelmän avulla, joten uudelleen käynnistyessään oliot voivat palauttaa aiemmat asetuk-

sensa ja jatkaa toimintaansa normaalisti. Mikäli haluttua varmistettua tietoa ei löydy, objektit käyttävät niille ominaisia oletusarvoja tai raportoivat virhetilanteesta managerilleen ja jäävät odottamaan uusia käskyjä.

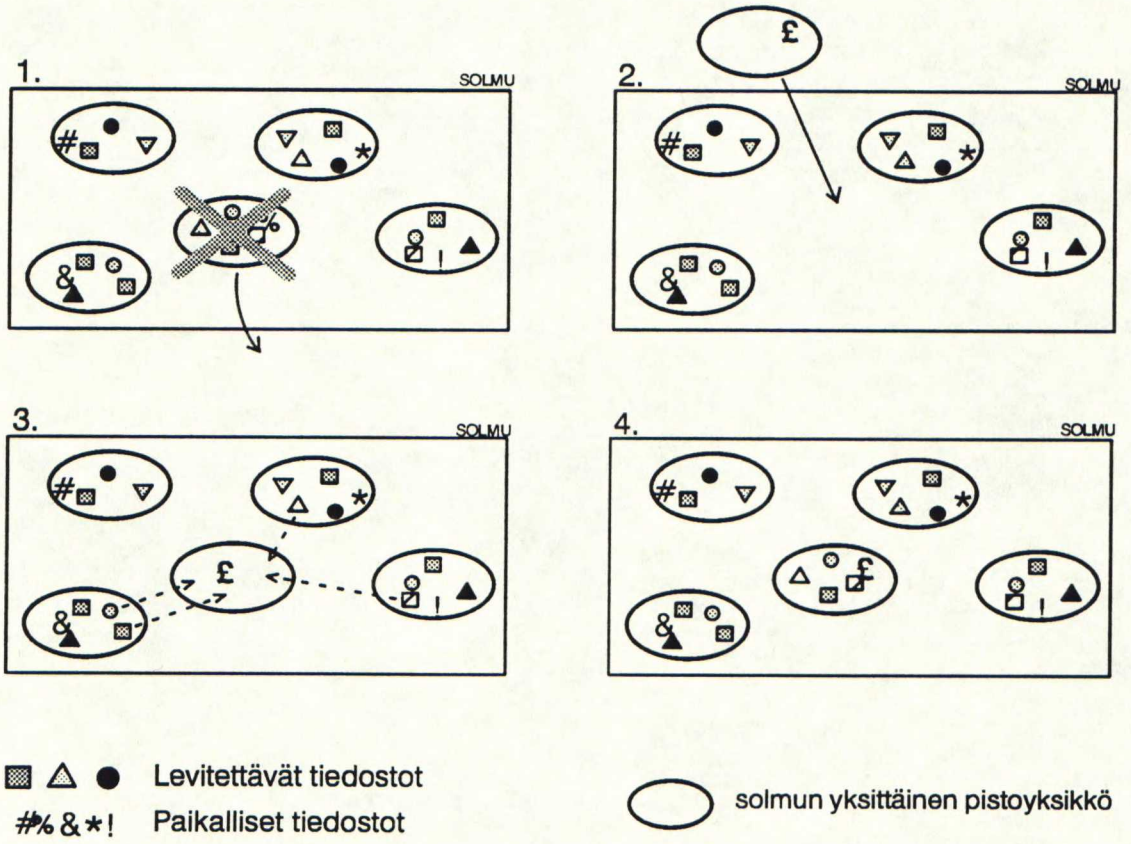


Kuva 8 Solmun sisältämä ohjelmisto ja sen rajapinnat

Yksinkertainen omalle pistoyksikölle tapahtuva tiedon varmistaminen ei riitä niissä tilanteissa, joissa pistoyksikkö vioittuu ja se joudutaan korvaamaan uudella yksiköllä. Uuden pistoyksikön saadessa käyttöjännitteen, sen prosessori käynnistää yksikön objektit, mutta tällöin omalta yksiköltä löytyvät asetukset eivät kuulu uuteen toimintaympäristöön ja syntyy konflikti. Tämän välttämiseksi tiedonvarmistuspalvelun tulee kopioida tallennetut tiedot myös solmun jollekin toiselle pistoyksikölle (kuva 9), jolloin on mahdollista palauttaa uuden yksikön objekteille viimeisimmät uuteen ympäristöön kuuluvat tilatiedot (kuva 10). Redundanssin suhteen joudutaan käytännön syistä tekemään kompromissiratkaisuja ja on yhä mahdollista, että objekti ei löydä ympäristöön sopivia asetuksia ja joutuu pyytämään apua managerilta.



Kuva 9 *Paikalliset ja solmun sisällä levitettävät tiedostot*



Kuva 10 *Solmun pistoyksikön korvaaminen toisella*

Jos jokin olio on käyttökelvoton merkittävän ajan esimerkiksi pistoyksikön vioittuttua, saattaa sovellusohjelmisto joutua siirtämään kadonneen olion tehtävät muille

olioille. Käytännössä ei ole mitään mahdollisuutta päästä käsiksi viallisen pistoyksikön tietovarastoihin, jos yksikkö on niin huonossa kunnossa ettei se pysty suorittamaan ohjelmakoodiaan. Ratkaisun tarjoaa edellä mainittujen mahdollisten kopioiden löytyminen muilta pistoyksiköiltä, sillä sovellusohjelmisto voi saada tiedonvarmistusjärjestelmältä kopion kadonneen olion asetuksista.

Sovellusohjelmisto suorittaa transmissioon liittyvät toiminteet (ristikytkennän muutokset matalalla tasolla, varmennettujen yksiköiden tehtävien vaihto, siirtovirheseuranta jne.) pääosin käskyttämällä solmutoimintakerrosta, joka toimii sen agenttina /12/. Solmutoimintakerroksen käsittelemät tiedot liittyvät yleensä yhteen nimenomaiseen – omaan – pistoyksikköön, joten niitä ei saa levittää muille yksiköille; muutoin voisi käydä niin, että viallisen yksikön korvaava uusi yksikkö löytäisi esimerkiksi entisen yksikön laserin viritysarvot, ottaisi ne käyttöön omiensa sijasta ja transmissio häiriintyisi.

Koska tiedonvarmistusjärjestelmällä ei ole mahdollisuutta tulkita sille varmistettavaksi annetun tiedon sisältöä, täytyy palvelun käyttäjän määrätä, onko varmistettava tieto sellaista, että se säilytetään vain omalla yksiköllä, vai varmistetaanko se oman yksikön ulkopuolelle.

Vaikka ristikytkentä ohjelmallisesti suoritetaankin pistoyksikkötason solmutoimintakerroksessa, ristikytkentätiedot kuvaavat solmulle asetettuja toimintavaatimuksia, joten niiden varmistaminen kuuluu sovellusohjelmiston tehtäviin. Tiedonvarmistuspalvelun kannalta ristikytkentätaulukoiden käsittely ei poikkea muista varmistustoimenpiteistä – palvelun pyytäjänä on sovellusolio eikä varmistettavan tiedon sisällöllä ole varmistuspalvelun kannalta merkitystä – mutta ristikytkentätaulukoiden suuri koko on palvelun kannalta huomioon otettava tekijä.

3 VAATIMUSMÄÄRITTELY

3.1 Johdanto

Tiedonvarmistuspalvelun tarkoituksena on ensisijaisesti mahdollistaa nopea solmun käynnistyminen sähkökatkon tai hallitun alasajon jälkeen ja parantaa verkonhallinnan luotettavuutta monistamalla tärkeitä tietoja.

Tiedonvarmistusjärjestelmä toteutetaan tiedostopohjaisena palveluna. Solmun toiminnan kannalta on perusteltua jakaa palvelut kahteen ryhmään; osa tiedoista liittyy yksittäiseen pistoyksikköön (mm. laserlähettimen ja oskillaattorien viritysparametrit sekä erilaiset yksikkötunnisteet) ja osa on yleisemmän tason tietoa, joka liittyy pistoyksikön yleiseen rooliin solmun osana (mm. hälytyskonfiguraatiot, salasanat ja ristikytkentätaulukot).

Fyysiset tiedostot jaetaan solmun pistoyksiköille siten, että paikalliset tiedot talletetaan ainoastaan omalle yksikölle, mutta yleisemmän tason tiedot, nk. levitettävät tiedot, voidaan kopioida eri puolille fyysistä solmua. Levitettävistä tiedostoista on varmistuspalvelua pyytäneellä pistoyksiköllä aina yksi kopio ja tästä alkuperäisestä tiedostosta tehdään mahdollisuuksien mukaan kopio jollekin toiselle solmun pistoyksikölle; tällöin yksittäinen vikaantunut pistoyksikkö voidaan poistaa solmusta siten, että sen perusasetukset jäävät kopion muodossa solmuun odottamaan korvaavaa pistoyksikköä. Kun uusi yksikkö asennetaan vanhan paikalle, tiedonvarmistusjärjestelmä palauttaa solmun eri pistoyksiköiltä löytyvät vanhan yksikön tiedostot uudelle yksikölle, joka jatkaa toimintaa vanhan yksikön roolissa.

Alkuperäisestä tiedostosta tehtävien kopioiden määrä on rajoitettu käytössä olevien muistiresurssien vähyyden vuoksi yhteen.

Kahden tiedostotyypin käsittely eroaa siirrettäessä tiedoston sisältävä pistoyksikkö solmusta toiseen: paikalliset tiedostot säilyvät muuttumattomina, mutta levitettävät tiedostot häviävät automaattisesti, jotta vältytään sekaannuksilta.

Solmun käynnistyessä kaikki varmistuksia tekevät ohjelmistokomponentit ovat lähes samanaikaisesti kiinnostuneita toimintaansa vaikuttavista varmistetuista tiedoista, joten tiedonvarmistusjärjestelmän toiminnan kannalta kriittisin osa on toi-

minta solmun käynnistyessä. Toinen kuormitustilanne syntyy, kun solmun toiminta konfiguroidaan toisentyyppiseksi, esimerkiksi päätekanavointilaite muutetaan toistimeksi. Näihin tilanteisiin on varauduttava varmistusjärjestelmän suunnitteluvaiheessa, jotta tukipalvelu ei muutu solmun toiminnan kannalta keskeisempien toiminteiden esteeksi eikä aiheuta palvelun asiakkaina toimiville solmun muille ohjelmistokomponenteille kohtuutonta hidastetta.

3.2 Käyttöliittymän yleinen määrittely

Tiedonvarmistuspalvelun käyttöliittymänä toimii C++ –kielinen rajapintaluokka, jonka kukin asiakkaana toimiva ohjelmistokomponentti sisällyttää omaan prosessiinsa. Tiedostojen käsittely tapahtuu tämän luokan metodien kautta. Paikallisten ja levitettävien tiedostojen käsittelyyn rajapinta tarjoaa eri metodit, jotta tiedostotyyppien välillä ei synny sekaannuksia.

Levitettyjen tiedostojen käsittely pistoyksiköiden välillä on palvelun käyttäjän kannalta piilossa, joten asiakkaat näkevät paikallisten ja levitettävien tiedostojen käsittelyn samanlaisina, paitsi vasteajan suhteen, joka levitettävien tiedostojen kyseessä ollen voi olla pidempi kuin paikallisilla tiedostoilla.

Asiakkaat näkevät palvelun tiedostoina, joita voi luoda, päivittää, lukea ja hävittää. Tiedonvarmistuspalvelu ei rajoita asiakkaiden käyttämiä tiedostonimiä. Ristiriitojen välttämiseksi asiakkaiden käyttöön on ylemmällä järjestelmätasolla määrätty nimiryhmät.

Tarjotun palvelun tulee datan suhteen olla läpinäkyvä eikä tiedonvarmistusjärjestelmä saa missään vaiheessa manipuloida tiedoston sisältöä.

Käyttöliittymänä toimivan rajapintaluokan on kestettävä virheellinen käyttö ja estettävä sen vaikutukset muiden asiakkaiden saamiin palveluihin.

Mikäli jokin ulkoinen virhe estää tiedonvarmistusjärjestelmää tarjoamasta luvattuja palveluita, asiakkaille ilmoitetaan, ettei palvelua ole saatavissa. Samalla ryhdytään toimenpiteisiin virheen vaikutuksen eliminoimiseksi ja normaaliin toimintaan palaamiseksi.

Käytännössä tiedonvarmistuspalvelun käyttö asiakkaan kannalta etenee solmun käynnistymisen ja jännitteen katkaisemisen välillä kahdessa vaiheessa:

Vaihe 1: Käynnistyksen yhteydessä haetaan mahdollisesti olemassaolevat varmistustiedostot. Jos tiedostoja ei ole, käytetään vastaavien parametrien ja asetusten oletusarvoja tai kysytään arvot olion managerilta.

Vaihe 2: Muuttuneet kriittiset tiedot päivitetään ajonaikaisesti tiedostoon heti muutoksen jälkeen.

Asiakkaiden on syytä sisällyttää varmistamaansa tietoon tunniste, josta ne erottavat omien eri ohjelmistoversioiden tekemät varmistukset toisistaan. Osa varmistettavista tiedoista on luonteeltaan sellaisia, ettei niitä muuteta vuosikausiin, joten ohjelmistopäivityksen myötä pistoyksikölle tulevan uuden ohjelmaversion täytyy voida erottaa aiempien ohjelmien mahdollisesti erilaiset tietorakenteet uusista.

3.3 Käyttöliittymän metoditason kuvaus

Seuraavassa esitellään käyttöliittymärajapinnalta vaadittavat metodit parametreineen. Parametreistä on yhteenveto heti metodikuvausten jälkeen.

Koska palvelun vasteajat myöhemmin esitettävistä laitteistoon liittyvistä syistä ovat ennustamattomat ja ennen kaikkea mahdollisesti asiakkaiden kannalta katsottuna suhteellisen pitkiä – jopa useita sekunteja – palvelupyynnöt on mahdollista esittää joko synkronisina tai asynkronisina.

Synkronisessa tilassa ohjelman kontrolli palaa asiakkaalle vasta kun palvelupyyntö on käsitelty, eli asiakas jää odottamaan palvelun toteutumista. Reaaliaikajärjestelmässä tämä tarkoittaa sitä, että asiakas luopuu prosessointiajastaan muiden ohjelmistokomponenttien hyväksi kunnes haluttu varmistuspalvelu on suoritettu.

Asynkronisessa tilassa rajapintakutsu aiheuttaa palvelupyynnön siirtymisen tiedonvarmistusjärjestelmän jonoon, mutta asiakas pääsee jatkamaan muita tehtäviään välittömästi. Tiedonvarmistusjärjestelmän saatua riittävästi prosessointiaikaa ja hoidettua vaaditut tiedosto–operaatiot se lähettää asiakkaalle IUCM–sanoman (*Inter–unit Communication Manager*), jonka vastaanotettuaan asiakas käy tarkasta-

massa toiminnan tuloksen käyttöliittymän rajapinnasta. IUCM on pistoyksiköiden välisiä sanomapalveluja tarjoava solmun ohjelmistokomponentti. IUCM–sanomaa käytetään tavallisen käyttöjärjestelmäsignaalin sijasta siksi, että useimmilla palveltavilla ohjelmistokomponenteilla on tehtäviä, jotka edellyttävät pistoyksiköiden välistä liikennöintiä, joten ne viettävät suuren osan ajasta vastaanottaen ja tulkiten IUCM–sanomia. Täten tiedonvarmistuspalvelun lähettämä sanoma tulee nopeasti käsittelyyn.

Palvelupyynnön tilan seuranta

Asiakas kysyy jokaisen esittämänsä palvelupyynnön lopputuloksen kutsuen `getStatus()` –metodia.

Uusi palvelupyyntö voidaan esittää vasta, kun edellinen pyyntö on käsitelty ja lopputulos kysytty. Tällä järjestelmätason päätöksellä varmistetaan, että mahdollisesti häiriötilanteessa villiintyvä asiakas ei pääse kuluttamaan kaikkia pistoyksikön tiedonvarmistusresursseja eikä polttamaan yksikön EEPROM–muistia käyttökelvottomaksi.

```
getStatus( BAMStatusType &returnStatus )
```

Paluuarvo kertoo joko edellisen palvelupyynnön tuloksen tai asynkronisen toiminnan tapauksessa mahdollisesti, että palvelupyynnön käsittely on vielä kesken.

Tiedoston luominen

`create...File()` –komennolla asiakas pyytää uuden tiedoston luomista. Paikallisia ja levitettäviä tiedostoja varten on oma metodinsa:

```
createLocalFile( BAMFileNameType name, BAMFileSizeType
size, BAMModeType mode )

createDistributableFile( BAMFileNameType name, BAMFile-
SizeType size, BAMModeType mode )
```

Tiedonvarmistuspalvelun käyttäjä voi kutsua tätä metodia käynnistysvaiheessa varatakseen itselleen tiedostokapasiteettia, jos se ei löydä varmistettua tiedostoa, tai

myöhemmin aina ennen tiedoston päivitystä varmistaakseen, että tiedosto on olemassa ennen päivitysyritystä.

Tiedoston nimi on ainutkertainen pistoyksikön sisällä eli järjestelmä ei anna luoda kahta tiedostoa samalla nimellä. Paikallisten ja levitettävien tiedostojen nimet ovat kuitenkin toisistaan riippumattomia.

`mode`–parametrilla määrätään, halutaanko palvelu synkronisena vai asynkronisena.

Tiedon varmistaminen tiedostoon

Tieto päivitetään käyttäjän allokoimasta tavutaulukosta tiedostoon `write()`–metodilla. Paikallisille ja levitettäville tiedostoille on jälleen omat metodinsa.

```
writeLocal( BAMFileNameType name, BAMFileSizeType size,
BYTE *source, BAMModeType mode )
```

```
writeDistributable( BAMFileNameType name, BAMFileSizeType
size, BYTE *source, BAMModeType mode )
```

`source`–parametri on osoitin varmistettavaan tavutaulukkoon.

Tiedon palauttaminen tiedostosta

Tieto palautetaan asiakkaan allokoimaan tavutaulukkoon kahdessa vaiheessa johtuen asiakkaiden dynaamisesta toiminnasta tiedonvarmistusjärjestelmään verrattuna: ensimmäisessä vaiheessa tiedoston sisältö noudetaan rajapintaluokkaan (operaatio kestää muutamista kymmenistä millisekunneista sekunteihin) ja toisessa vaiheessa noudettu data siirretään eri käskystä asiakkaalle. Paikallisille ja levitettäville tiedostoille on ensimmäistä vaihetta varten omat metodinsa.

```
fetchLocal( BAMFileNameType name, BAMFileSizeType size,
BAMModeType mode )
```

```
fetchDistributable( BAMFileNameType name, BAMFileSizeType
size, BAMModeType mode )
```

Toinen vaihe suoritetaan kutsumalla paikallisille ja levitettäville tiedostoille yhteistä metodia `readFetchedFile()`, jonka parametrina annetaan osoitin asiakkaan tavutaulukkoon:

```
readFetchedFile( BYTE *destination )
```

`readFetchedFile()` –metodin jälkeen ei tarvitse kutsua `getStatus()` –metodia, sillä palautus tapahtuu aina nopeasti eikä voi epäonnistua, jos tietojen noutaminen rajapintaan on onnistunut.

Muun kuin `readFetchedFile()` –metodin kutsuminen `fetch...()` –metodien jälkeen johtaa tiedoston tiedonvarmistusjärjestelmän sisäisällä tapahtuvaan siirtoon käytettävän puskurin ylikirjoittumiseen ja noudetun tiedon muuttumiseen asiakkaan kannalta käyttökelvottomaksi.

Tiedoston tuhoaminen

Asiakkaat voivat tuhota tarpeettomiksi käyneitä tiedostoja yksi kerrallaan metodilla `delete...File()`. Metodia tarvitaan pääsääntöisesti silloin, kun olemassa olevan tiedoston pituutta halutaan muuttaa (tapahtuu hävittämällä tiedosto ja luomalla se uudestaan uudella pituudella), tai kun varmistettava tieto käy lopullisesti tarpeettomaksi.

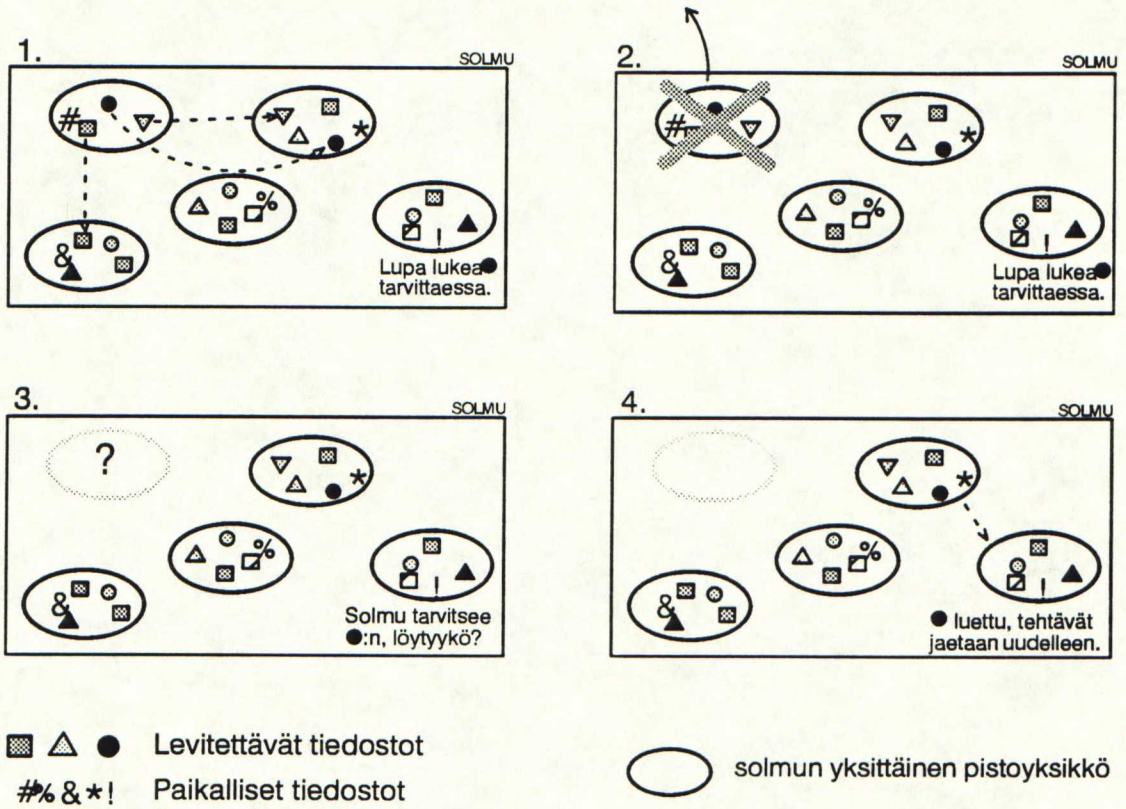
```
deleteLocalFile( BAMFileNameType name, BAMModeType mode )

deleteDistributableFile( BAMFileNameType name, BAMMode-
Type mode )
```

Levitykseen määrätyn tiedoston tuhoaminen aiheuttaa kaikkien kyseisestä tiedostosta solmussa olevien kopioiden hävittämisen, mutta asiakas saa tiedon operaation päättymisestä heti kun omalla pistoyksiköllä sijainnut kopio on tuhottu. Näin siksi, että suuren kuormituksen aikana saattaa kestää pitkään, ennenkuin hävityskäsky on levinnyt solmun kaikille yksiköille ja ehditty käsitellä niillä.

Tiedosto häviää vain solmussa läsnä olevilta yksiköiltä; jos joku yksikkö on hyllyssä varakorttina, sillä mahdollisesti sijaitseva tiedostokopio voi myöhemmin aiheuttaa hämmennystä. Siksi onkin suositeltavaa tiedoston sisäisen rakenteen muuttuessa luoda uusi rakenne toisella nimellä.

Varjokopioiden lukeminen

Tietyissä tilanteissa on tarpeen, että millä tahansa pistoyksiköllä sijaitseva asiakas voi lukea solmusta poistuneen yksikön jälkeensä jättämiä nk. varjotiedostoja (kuva 11). Normaalit tiedostojen lukemiseen tarkoitetut metodit antavat tarkoituksellisesti lukea vain omalta yksiköltä luotujen tiedostojen sisältöjä, joten varjokopioiden lukemiseen tarvitaan omat metodinsa. Varjotiedoston sisällön muuttaminen ei ole sallittua.



Kuva 11 Varjotiedoston syntyminen ja sisällön lukeminen

Ensimmäisenä askeleena asiakkaiden normaalisti käyttämä tiedostonimi on muutettava tiedonvarmistusjärjestelmän sisäisesti käyttämään täydellisempään muotoon. C++:n tyyppimäärittelyn vuoksi asiakkaan ei tarvitse tietää 'pitkän nimen' sisäistä rakennetta. Muunnos tapahtuu kutsumalla metodia

```
convertIntoLongIdentifier( BAMFileNameType name, BAMLong-
IdentifierType &returnValue )
```

Kyseessä on pelkkä muunnos eikä tiedoston olemassaoloa tarkisteta.

Muunnos on suoritettava sillä yksiköllä, jonka varjotiedoston käsittelystä on kyse. Muunnoksen tekijä toimittaa saadun tunnisteen IUCM:n avulla sille toisella pistoyksiköllä sijaitsevalle asiakkaalle, jolle se haluaa antaa pääsyn tiedostoihinsa. Näin varmistuu, että kuka tahansa ei pääse käsiksi toisen yksikön tiedostoihin.

Pitkän tiedostotunnisteen saatuaan mikä tahansa solmun tiedonvarmistusjärjestelmän asiakas voi lukea tiedoston sisällön metodilla `fetchAShadow()`. Tämä komento vastaa edellä esiteltyjä `fetch...File()` –metodeja eli onnistuneen noutamisen jälkeen tieto siirretään palvelurajapinnan yli `readFetchedFile()` –metodilla.

```
fetchAShadow( BAMLongIdentifierType file_specifier, BAM-
FileSizeType size, BAMModeType mode )
```

Yhteenveto metodien yhteydessä käytetyistä parametreista:

`BAMLongIdentifierType file_specifier`
: tunniste, jonka avulla tiedonvarmistusjärjestelmä yksikäsitteisesti tunnistaa määrätyn tiedoston koko solmusta

`BYTE *destination`
: osoitin tavumuotoiseen taulukkoon, johon tieto halutaan siirtää

`BYTE *source`
: osoitin tavumuotoiseen taulukkoon, josta tieto halutaan siirrettäväksi

`BAMFileSizeType size`
: tiedoston pituus, yksikkönä tavu

`BAMFileNameType name`
: tiedonvarmistusjärjestelmän tiedostoniminä tuntemasta lukujoukosta valittu tunniste, jonka järjestelmä liittää asiakkaan haluamaan tiedostoon sen luomisen yhteydessä

`BAMModeType mode`
: määrää koska ohjelmakontrolli palaa kutsutusta rajapintametodista takaisin asiakkaalle (`bamSynchronous` | `bamAsynchronous`)

`BAMStatusType returnStatus`
: kertoo edeltäneen palvelupyynnön tilan; jos yhtään pyyntöä ei vielä ole esitetty, arvo on `no_requests`; jos pyynnön prosessointi on kesken, arvo on `not_completed`, ja muut arvot edeltäneestä loppuun saakka prosessoidusta pyynnöstä ovat palvelupyynnöstä riippuen:

pyyntö:	arvo:
"luo"	`ok`, `duplicate_name`,

```
                error,
                service_unavailable.

"varmista"      ok,
                too_large_size_given,
                error,
                service_unavailable.

"nouda"         ok,
                file_not_found,
                too_large_size_given,
                error,
                service_unavailable.

"hävitä"        ok,
                error,
                service_unavailable.
```

3.4 Laitteiston asettamat rajoitukset

Vasteaika palvelun pyytämishetkestä vaaditun toiminnan suorituksen päättymiseen vaihtelee riippuen solmun ja yksittäisten pistoyksiköiden kuormituksesta. Vasteaikojen vaihtelu johtuu suurelta osin siitä, että tiedostot matalalla tasolla talletetaan hitaasti kirjoittuvalle EEPROM:lle, eikä useiden tiedostojen samanaikainen käsittely ole muistiajurin rajoitusten takia mahdollista. Lisäksi asiakkaat kuormittavat palvelua reaaliaikakäyttöjärjestelmästä johtuen toisistaan riippumattomasti, joten järjestelmän täytyy sisältää asiakkailta piilotettu palvelujono.

Tiedon lukemiseen EEPROM–piiriltä kuluva aika on käytännöllisesti katsoen merkityksetön, mutta muistiin kirjoittaminen on hidasta (10–20 ms/tavu) /13//14/, joten lyhytkin tiedoston päivitys kuluttaa ajallisia resursseja paljon lisäten kuormitushuipun mahdollisuutta ja kasvattaen käyttäjien näkemiä vasteaikoja.

EEPROM:n rajallisten kirjoituskertojen (luokkaa $10^4 - 10^5$ kirjoitusta/tavu) vuoksi palvelua ei saa käyttää usein muuttuvien tietojen varmistamiseen. Pistoyksiköiden välisessä levitettävien tiedostojen käsittelyssä tiedonvarmistusjärjestelmä ottaa tämän muistipiirien elinikään vaikuttavan rajoituksen huomioon, mutta ylemmällä tasolla vastuu on asiakkaiden harteilla, sillä tiedonvarmistuspalvelun on oltava tarjolla kaikille asiakkaille eikä se voi diskriminoida palvelupyyntöjä. Käytännössä ongelma on ratkaistu järjestelmätason päätöksellä, joka määrää mitä tietoja kukin ohjelmistokomponentti tallettaa ja kuinka usein.

Kullakin pistoyksiköllä olevan EEPROM:n määrä voi aiheuttaa rajoituksia yhtä aikaa olemassa olevien tiedostojen määrälle tai koolle. Pienet rajoitteet pyritään kuitenkin kiertämään ohjelmallisesti aina kun mahdollista.

Luotettavan toiminnan kannalta olisi suotavaa, että tiedoston päivitys tapahtuisi luomalla päivitystä varten uusi tiedosto, tallentamalla tiedot siihen, nimeämällä uusi tiedosto vanhaksi ja poistamalla vanha versio, mutta käytännössä joudutaan tyytymään suoraan päällekirjoitukseen, jotta EEPROM:n kuluminen voidaan pitää minimissä ja koska muistiin kirjoittaminen on hidasta.

3.5 Yleiset vaatimukset

- Prosessointiajan käyttö

Tiedonvarmistusjärjestelmä ei missään oloissa saa viedä yhtäjaksoisesti niin paljon prosessointiaikaa, että pistoyksikön transmissioon tai verkonhallintaan liittyvät perustoiminteet häiriintyvät.

- Solmun hallitsemattoman alasajon jälkeinen toiminta

Solmun käynnistyessä uudelleen tutkitaan ensin solmun identiteetti. Jos haihtumattomasta muistista löytyvät levitettäviksi merkityt tiedostot eivät kuulu nykyiseen ympäristöön, ne hävitetään heti. Ympäristöön kuuluvat vastaavat tiedostot päivitetään ajantasaisimmiksi solmusta löytyviksi versioiksi ja annetaan asiakkaille vasta sen jälkeen. Paikallisiksi määrätyt tiedostot ovat aina asiakkaiden saatavilla.

- Käyttäjien tunnistus

Minkä tahansa asiakasohjelmiston tulee päästä käsiksi mihin tahansa oman pistoyksikön asiakkaan tiedostoon, jos sillä vain on esittää oikea tiedostonimi.

Hallitsemattomasti solmun toiminnasta poistuneen pistoyksikön tehtävien siirtämiseksi uudelle yksikölle on oltava mahdollista myös päästä lukemaan minkä tahansa solmussa sijainneen asiakkaan mitä tahansa tiedostoa (nk. varjotiedostojen käsittely). Tämä palvelu on tarkoitettu rajoitetulle solmusovellusohjelmistolle ja edellyt-

tää tavallista tarkemman tiedostotunnisteen esittämistä. Mikään vieras käyttäjä ei kuitenkaan voi poistaa mainittua varjotiedostoa, eikä muuttaa sen sisältöä.

- Sisäinen toiminta ja luotettavuus

Mahdollinen tiedoston kopiointi vieraalle pistoyksikölle aloitetaan vasta kun omalla yksiköllä sijaitseva kopio on päivitetty.

Mikäli EEPROM–ajurina toimiva matalan tason tiedostopalvelut tiedonvarmistusjärjestelmälle tarjoava MeM (*Memory Manager*) ilmaisee ylläpitämänsä tarkistussumman perusteella jonkun tiedoston olevan viallinen /15/, tiedosto hävitetään automaattisesti ja asiakkaalle kerrotaan, ettei tiedostoa löydy. EEPROM–piirien valmistajat lupaavat kertaalleen kirjoitetun tiedon pysyvän muuttumattomana kymmenen vuotta /13//14/, mutta myös kesken tiedoston päivityksen tapahtuva jännitekatko aiheuttaa tiedoston vioittumisen.

- Samanaikaisten käyttäjien lukumäärä

Asiakasolioiden määrä on rajoittamaton. Asiakkaiden lukumäärä yhdessä käyttöjärjestelmäprosessissa rajoitetaan kuormitustilanteiden minimoimiseksi kymmeneen.

- Herkkyys ulkopuolisten tukikomponenttien häiriöille

Suunnittelussa tulee pyrkiä minimoimaan tukikomponenttien (MeM ja IUCM) vastenopeuksien vaikutus järjestelmän toimintaan.

- Testattavuus

Paikallisten tiedostojen käsittelyn testaaminen on yksinkertaista, mutta muuttuvaan ympäristöön sopeutuvan tiedonvarmistusjärjestelmän testaaminen levitettävien tiedostojen ja suorituskyvyn osalta on mutkikasta. Suunnittelussa on kiinnitettävä huomiota solmutason ratkaisujen yksinkertaisuuteen ja loogisen kokonaisuuden ymmärrettävyyteen.

3.6 Ylläpidettävyys

- Laitteistomuutosten vaikutus

Kaikki laitteistoriippuvuudet peittyvät tiedonvarmistusjärjestelmän tukenaan käyttämiin muihin ohjelmistokomponentteihin, joten järjestelmä on näiltä osin riippumaton mahdollisista muutoksista. Mainittujen tukikomponenttien rajapinnan tai käytön muuttaminen aiheuttaa muutoksia tiedonvarmistusjärjestelmässä, joten niiden tulee olla hyvin perusteltuja.

Uusien pistoyksikkötyyppien esittely saattaa vaikuttaa järjestelmän päätöksentekologiikkaan.

- Käyttöliittymän muutokset

Käyttöliittymän tulee pysyä vakaana eli kerran esiteltyjä palveluita ja niiden käyttöä ei ilman painavia perusteita saa muuttaa, koska ne vaativat muutoksia asiakaskomponenteissa. Uusien palveluiden esittely on sallittua.

- Sisäiset muutokset

Järjestelmän sisäiset algoritmit, sanomaprotokollat, jonopalvelut ja työnjako ovat piilossa käyttäjiltä, joten niitä voidaan tarvittaessa muuttaa.

4 RATKAISTAVIA KYSYMYKSIÄ

4.1 Tiedon hajauttaminen ja versionhallinta

Palveltavan asiakkaan levitykseen määräämästä tiedostosta pyritään pitämään aina yksi kopio solmun jollakin toisella pistoyksiköllä. Ennen käytännön levitysstrategian päättämistä on ratkaistava eräitä ongelmia:

Solmun käynnistyessä ei ole heti mahdollista tietää, mitä yksiköitä solmuun kuuluu ja mitä tiedostoja kultakin yksiköltä löytyy, joten ensimmäinen tehtävä on kartoittaa tiedostotilanne.

Yksittäisen pistoyksikön käynnistyessä sen tulee kertoa muille solmusta löytyville yksiköille oma olemassaolonsa. On mahdollista, että solmu on koottu uusista pistoyksiköistä, eikä niitä ole vielä koskaan käytetty yhdessä, joten aluksi on päätettävä solmun identiteetti tiedonvarmistusjärjestelmän kannalta. Käytännössä tämä tarkoittaa sitä, että kukin pistoyksikkö tutkii ensin, onko se kuulunut osana johonkin solmuun ennen juuri tapahtunutta käynnistystä. Mikäli yksikkö on aivan uusi, se luo itselleen satunnaisen solmutunnisteen ja ilmoittaa sen muille yksiköille, muussa tapauksessa se ilmoittaa kuuluvansa osana ennalta määrättyyn solmuun.

Tiedonvarmistusjärjestelmään kuuluu osana jokaisella pistoyksiköllä sijaitseva logiikka, joka kerää yksiköiden raportoimat tunnisteet ja tekee niiden perusteella yksikkökohtaisen päätöksen koko solmun tiedonvarmistuksellisesta identiteetistä. Kullakin pistoyksiköllä tulee olla oma päätöksentekoelin, koska kaikkien yksiköiden on kyettävä tarvittaessa toimimaan riippumatta muiden yksiköiden järjestelmien toimintakyvystä. Monistetusta luonteesta johtuen kaikkien yksiköiden päätöksentekoelimet käyttävät samaa logiikkaa ja päätyvät samoihin tuloksiin olettaen että niillä on sama lähtötieto solmun kokoonpanosta eli läsnä olevista yksiköistä.

Kun pistoyksiköt ovat päättäneet solmutason identiteetin, ne automaattisesti hävittävät kaikki levitykseen määrätyt tiedostot, jotka eivät kuulu solmuun.

Seuraavassa vaiheessa kunkin pistoyksikön täytyy varmistua siitä, että omalta yksiköiltä löytyy ajantasaisin versio kaikista levitykseen määrätyistä tiedostoista. Tämä tapahtuu kyselemällä läsnä olevilta yksiköiltä, onko niillä kopioita omista levityk-

seen määrätyistä tiedostoista. Ongelmaksi nousee nyt kuitenkin ajantasaisuuden määrittely; kaikki nykyiseen solmukokoonpanoon kuulumattomat tiedostot on hävitetty, mutta mistä tiedetään, mikä mahdollisista useista erilaisista jäljelle jääneistä samannimisistä tiedostoista on ajantasaisin?

Looginen ratkaisu olisi liittää jokaiseen tiedostoon nimen lisäksi aikaleima; toteutusympäristönä toimiva SDH–solmu sisältää reaaliaikakellon, joten ajan saaminen on mahdollista, mutta kello on voinut jäädä asettamatta aikaan, aika on voitu asettaa väärin, tai sähkökatko on voinut nollata kellon. Ongelmia voivat aiheuttaa myös lähes samassa ajassa käyvät solmut, joiden välillä esimerkiksi asennustyön vuoksi vaihdellaan pistoyksiköitä ja joilla on sama identiteetti esimerkiksi siksi, että uuden solmun pistoyksiköt on opetettu tiettyihin toimintoihin käyttämällä niitä ensin vanhassa solmussa.

Luotettavimmaksi versiontunnistusratkaisuksi on valittu versionnumerointi, jossa versionumero kasvaa aina tiedostoon kirjoitettaessa. Tämä yhdessä solmutunnisteen kanssa takaavan riittävän luotettavuuden, vaikkei olekaan aukoton menetelmä.

Käytännössä yksittäisen pistoyksikön tiedonvarmistusjärjestelmä kyselee muilta yksiköiltä tuoreinta versiota tiedoston nimen ja siihen liitetyn versiotunnisteen avulla. Tunnistukseen on liitettävä myös tiedostopituus, jotta vältytään eri–ikäisten ohjelmaversioiden mahdollisesti aiheuttamilta sekaannuksilta. Kukin vieras yksikkö saadessaan kyselyn vastaa siihen suuremmalla versionumerolla, jos omaa tuoreemman version, tai jättää vastaamatta. Näin saadaan samalla pidettyä pistoyksiköt yhdistävän lähiverkon kuormitus tiedonvarmistuksen osalta mahdollisimman pienenä. Jos vieras yksikkö toteaa kysyjällä olevan uudemman version kyseisestä tiedostosta, se automaattisesti hävittää oman versionsa, joten vanhat versiot eivät jää kummittelemaan solmun vieraille pistoyksiköille, jos lähiverkkoyhteys on kunnossa. Mikäli vieraan yksikön versio on kysyjän versiota tuoreempi, vieras versio kopioidaan kyselyn tehneelle yksikölle samalla versionumerolla, joka sillä oli vieraalla yksiköllä.

4.2 Suorituskyky

Haihtumattomana muistina käytettävien EEPROM–piirien hidas kirjoittaminen nousee käynnistymisen yhteydessä suoritettavassa tiedostojen ajantasaistamisessa merkittäväksi pullonkaulaksi, sillä yhden tavun kirjoittaminen kestää muistipiiristä

riippuen 10–20 ms, jolloin esimerkiksi kilotavun kokoisen tiedoston kirjoittaminen pistoyksikön EEPROM–muistiin voi kestää toista kymmentä sekuntia. Toisaalta MeM huolehtii siitä, että vain tiedoston muuttuneet tavut ohjelmoidaan EEPROM:lle, mikä useimmissa tapauksissa nopeuttaa toimintaa. Koska muistia voi kerrallaan käsitellä vain yhden tiedoston osalta, kaikki käynnistyksen aikainen haihtumattomaan muistiin kirjoittaminen on kuitenkin minimoitava.

Tiedonvarmistusjärjestelmällä ei ole tietoa palveltavien asiakkaiden tiedostojen mahdollisista keskinäisistä riippuvuuksista ja tärkeyssuhteista, joten kaikkia tiedostoja kohdellaan samanarvoisina, eikä voida sallia tapausta, jossa taataan heti käynnistyksen jälkeen jonkun omalla pistoyksiköllä sijaitsevan tiedoston ajantasaisuus, mutta toisen tiedoston tuorein versio löytyisi vieraalta pistoyksiköltä ja kopioitaisi omalle yksikölle vasta myöhemmin; muussa tapauksessa voisi syntyä palveltavien asiakkaiden kannalta ylipääsemättömiä ongelmia esimerkiksi säännöllisesti pian käynnistyksen jälkeen solmun kaatavan vian sattuessa, tai kun asiakkaiden keskinäinen vuorovaikutus on riippuvainen varmistettujen tietojen saatavuudesta.

Solmun käynnistyessä annetaan asiakkaille pääsy paikallisiin tiedostoihin, päivitetään oman pistoyksikön alkuperäiset levitykseen määrätyt tiedostot ajantasaisiksi solmuun kuuluviksi versioksi ja lopuksi jäädytetään levitettyjen tiedostojen keskinäiset suhteet siten, että oman pistoyksikön tiedostoja pidetään tuoreimpia ja annetaan asiakkaille pääsy myös näihin tiedostoihin.

Tämän jälkeen ollaan normaalissa toimintatilassa, jossa kaikki lukupyynnöt kohdistuvat omalle pistoyksikölle talletettuihin tietoihin ja levitykseen määrättyjä tiedostoja kopioidaan vieraille yksiköille aina tarpeen mukaan.

Koska tiedonvarmistusjärjestelmän käynnistymisen yhteydessä ei voida välttää EEPROM–piireille kirjoittamista levitettyjen tiedostojen ajantasaistamisessa, täytyy määritellä palvelujen saatavuus siten, että EEPROM:iin kirjoittamista vaativiin operaatioihin verrattuna nopeat lukupyynnöt käsitellään ennen kirjoituspyyntöjä.

4.3 Solmun tiedonvarmistuksellisen identiteetin muuttuminen

Levitykseen määrättyjen tiedostojen kopiot sitovat solmun yksiköt yhteen; joidenkin yksiköiden poistuttua ja mahdollisesti uusien ilmaannuttua solmuun järjestelmän tulee sopeutua muuttuneeseen kokoonpanoon.

Yksittäisen vioittuneen yksikön epäkuntoon joutuminen tai vaihtaminen ei muuta solmun identiteettiä, mutta uusista pistoyksiköistä muodostetun solmun on saatava mahdollisimman ainutkertainen identiteetti.

On tarpeen määritellä raja sille, kuinka suuri muutos solmussa aiheuttaa sen identiteetin muuttumisen ja kaikkien levitettäviksi luotujen tiedostojen automaattisen katoamisen.

Yksinkertaisen ratkaisun löytämistä vaikeuttaa pistoyksiköiden vaihteleva määrä eri solmukonfiguraatioissa: päätekanavointilaite sisältää 3–8 pistoyksikköä, toistinsolmu 3–5 ja ristikytkentäsolmu enimmillään 18 yksikköä /6/. Identiteettipäätöksen pohjana on mielekästä pitää enemmistöpäätöstä; mikäli suurin osa yksiköistä toteaa kuuluvansa samaan kokonaisuuteen, vähemmistö mukautuu tähän identiteettiin.

Kolmen yksikön tapaus on sikäli ongelmallinen, että jo kahden yksikön samanaikainen vaihtuminen hävittää kaikki levitettävät tiedostot. Käytännössä voidaan yksiköt kuitenkin vaihtaa yksi kerrallaan solmun käyttöjännitteen ollessa päällä, jolloin tiedostot säilyvät, ja levitettävien tiedostojen kohtalo on yksiköiden vaihtajan valittavissa. Ristikytkentäsolmun tapauksessa enemmistöpäätös sallii aina esimerkiksi aikakytkentäyksiköiden tai 2 Mbit/s-yksiköiden vaihtamisen uusiin ilman solmun identiteetin muuttumista.

Käynnistystilanteessa saman solmun pistoyksiköt saattavat käynnistyä hieman eri aikaan, joten identiteettipäätöksen tekoon liittyvä sanomanvaihto ja ajoitus on pyrittävä sekä optimoimaan että suojaamaan yksiköiden käynnistyksen ajalliselta huojunnalta.

4.4 Hajautusstrategia

Palveltavan asiakkaan levitykseen määräämistä tiedostoista pyritään kustakin pitämään yksi kopio jollakin toisella saman solmun pistoyksiköllä. Lukumäärä yksi on valittu rajallisen varmistuskapasiteetin ja ennen kaikkea EEPROM–piirien hitaan ohjelmoimisen takia, suurempi lukumäärä aiheuttaisi kohtuuttomia ongelmia käynnistysvaiheessa.

Joissakin tapauksissa – esimerkiksi muistin loppumisen takia – on mahdollista, ettei edes yhtä kopiota pystytä tekemään, mutta yksittäisten pistoyksiköiden muistiongelmien kiertäminen on useinmiten mahdollista kopioimalla tiedosto jollekin vajaalle yksikölle.

Pistoyksikköä, jolla palvelun pyytäjä sijaitsee, kutsutaan tässä manageriksi ja yksiköitä, joille manageri kopioi asiakkaan levitykseen määräämiä tiedostoja, kutsutaan vastaavasti agenteiksi. Kullekin managerille määrätään ympäristön kokoonpanosta riippuen yksi tai useampia agentteja, joita manageri saa käyttää tiedostokopioiden kantajina. Tietyn agentin ja managerin välistä yhteyttä kutsutaan linkiksi.

Agentin ja managerin välinen suhde on tässä tapauksessa sikäli poikkeuksellinen, että manageri saattaa toimia samanaikaisesti oman agenttinsa agenttina; tiedostojen vaihto ei kuitenkaan aina ole vastavuoroista ja agentti voi kieltäytyä vastaanottamasta tiedostoa vedoten muistin vähyyteen. Agentilla on myös oikeus koska tahansa vapauttaa managerin antaman tiedoston viemä tila omaan käyttöönsä edellyttäen, että se ilmoittaa asiasta managerille.

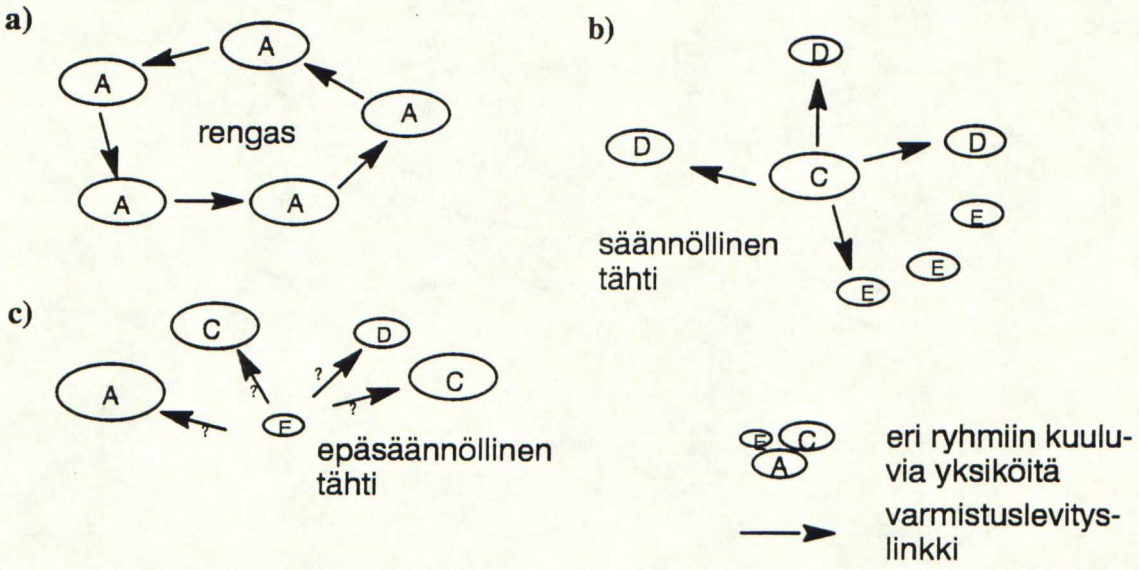
Solmun kaikki tiedonvarmistuksen piiriin kuuluvat pistoyksiköt on jaettu varmistusryhmiin, joiden sisällä tiedostojen levitys pyritään hoitamaan tiettyjen sääntöjen puitteissa. Ryhmät on nimetty A, B, C, ... Käytännön toteutuksessa tuetaan enintään viittä ryhmää. Kussakin ryhmässä saa olla yksiköitä 0–32 kappaletta ja jokainen pistoyksikkö kuuluu täsmälleen yhteen ryhmään.

Ryhmäjako tehdään ensisijaisesti EEPROM–muistin määrän mukaan siten, että paljon muistia omaavat yksiköt kuuluvat korkeisiin ryhmiin ja vähän muistia sisältävät yksiköt alempiarvoisiin ryhmiin. Ryhmiä vastaavia muistimääriä kuvaa sääntö $a \geq b \geq c \geq d \geq e > 0$, missä pienet kirjaimet vastaavat isoilla kirjaimilla määriteltyjen ryhmien muistikapasiteetteja.

Kunkin pistoyksikön oletetaan tarvitsevan omaan käyttöönsä enintään hieman yli puolet omasta kokonaismuistikapasiteetistaan; tällöin kahdelle toisiaan varmistavalle kortille voidaan aina taata mahdollisuus tiedoston varmistamiseen toiselle yksikölle. Tarkkaa omaan käyttöön varattavan muistin määrää ei pystytä etukäteen sanomaan, koska se vaihtelee tilanteen mukaan – joillakin yksiköillä suurin osa tiedosta on talletettu paikallisiin tiedostoihin, jolloin ne tarvitsevat vain vähän vierasta muistikapasiteettia.

Kun pistoyksiköitä on paikalla vain muutamia ja ne kuuluvat eri ryhmiin, tiedostojen levityksen onnistumistodennäköisyys on pienempi kuin jos yksiköt kuuluisivat samaan ryhmään.

Solmussa esiintyy kolmenlaisia varmistusrakenteita:



Kuva 12 Tiedostolevityksen perusrakenteet

Rengas on ketjurakenne, jossa yksiköt muodostavat päättymättömän suunnallisen renkaan. Kullakin yksiköllä on kaksi naapuria (jotka kahden yksikön tapauksessa ovat yksi ja sama yksikkö) – edeltäjä ja seuraaja. Kukin yksikkö käyttää pääsääntöisesti seuraajaansa tiedostokopioiden säilyttäjänä, mutta seuraajan ollessa kykenemätön tallentamaan kopion, voi manageri käyttää agenttinaan myös edeltäjäänsä (kuva 12a).

Säännöllisessä tähdessä manageri käyttää tiettyjä muita yksiköitä agentteinaan (kuva 12b). Manageri toimii keskusyksikkönä, jolla on satelliitteja. Keskusyksiköllä

on pääsääntöisesti enemmän muistia kuin yksittäisillä satelliiteilla, ja ideaalitapauksissa satelliittien vapaan muistin summa riittää täsmälleen keskusyksikön tarpeiden täyttämiseen. Manageri pyrkii kuormittamaan agenttejaan mahdollisimman tasaisesti, jotta muistin loppumisesta johtuvat palvelun vasteaikaa kasvattavat kieltäytymiset jäisivät vähäisiksi.

Epäsäännöllisessä tähdessä (kuva 12c) keskusyksikkö voi käyttää agentteinaan mitä tahansa muita yksiköitä riippumatta siitä, mihin ryhmään ne kuuluvat. Tiedostonkopiointitarpeen tullen manageri tarjoaa tiedostoa jollekin agentille, joka joko hyväksyy tai hylkää sen. Jos vastaus oli kielteinen, manageri yrittää tiettyyn rajaan asti löytää uuden agentin. Tämä varmistusrakenne on käytöksen ja suorituskyvyn kannalta epämääräinen ja sitä käytetään vain jos muut rakenteet eivät syystä tai toisesta ole mahdollisia.

Pistoyksiköiden tiedonvarmistusjärjestelmien on oltava riippumattomia toisistaan ja siten myös kunkin yksikön päätöksentekoelimet toimivat itsenäisesti. Koska varmistusrakenteiden muodostamispäätökset tehdään kullakin yksiköllä erikseen samojen sääntöjen pohjalta, eri yksiköiden päättämät strategiat ovat sopusoinnussa keskenään edellyttäen, että päätöksenteon perusteena käytetty tieto muista läsnä olevista yksiköistä on sama. Toteutuksessa lähdetään siitä, että lähtötiedot ovat luotettavia.

Kun solmuun lisätään tai siitä poistetaan pistoyksiköitä, joudutaan sovittua varmistusrakennetta (topologiaa) muuttamaan. Muuttunut rakenne rakenne on voimassa vasta päätöksentekohetken jälkeen tehtyjen tiedostolevitysten osalta, mikä voi johtaa siihen, että muuttuneen solmukonfiguraation ja uuden topologiapäätöksen välisenä aikana levitetystä tiedostosta ei pystytä tekemään oman yksikön ulkopuolella yhtään kopiota ja kopio syntyy uuden topologiapäätöksen jälkeen vasta tiedoston uudelleen päivittämisen yhteydessä. Tämän mahdollisuuden vaikutukset pyritään minimoimaan myöhemmin esiteltävän täydennyslevitysajastimen avulla.

Agenttien etsimisessä sovelletaan seuraavia sääntöjä alkaen korkeimmasta ryhmästä ja edeten alempiin ryhmiin, kunnes jokaisella yksiköllä on ainakin yksi nimetty agentti. Koska pistoyksiköiden tiedonvarmistuksen päätöksentekoelimillä ei ole keskinäistä vuorovaikutusta, agenttien valintaan vaikuttavat pistoyksiköiden paikat siten, että agentteja lähdetään varaamaan vasemmalta oikealle ja asennuskehikon oikeasta laidasta hypätään jälleen kehikon vasempaan laitaan. Kullakin kierroksel-

la voidaan varata agenteiksi ketjuihin ja säännöllisiin tähtiin vain vielä vapaita pistoyksiköitä.

Sääntö 1. Jos solmussa on useampia kuin yksi käsiteltävän ryhmän vielä vapaa yksikkö, yksiköt muodostavat renkaan.

Sääntö 2. Jos solmussa on vain yksi käsiteltävän ryhmän vielä vapaa yksikkö ja jos solmussa

Sääntö 2.1. on olemassa alempiin ryhmiin kuuluvia vielä vapaita yksiköitä, muodostetaan säännöllinen tähti, jonka keskusyksikkönä toimii edellä mainittu yksinäinen yksikkö. Satelliittien määrä riippuu käytettävissä olevista yksiköistä ja muistisuhteita. Satelliittien poiminta aloitetaan alimmasta ryhmästä.

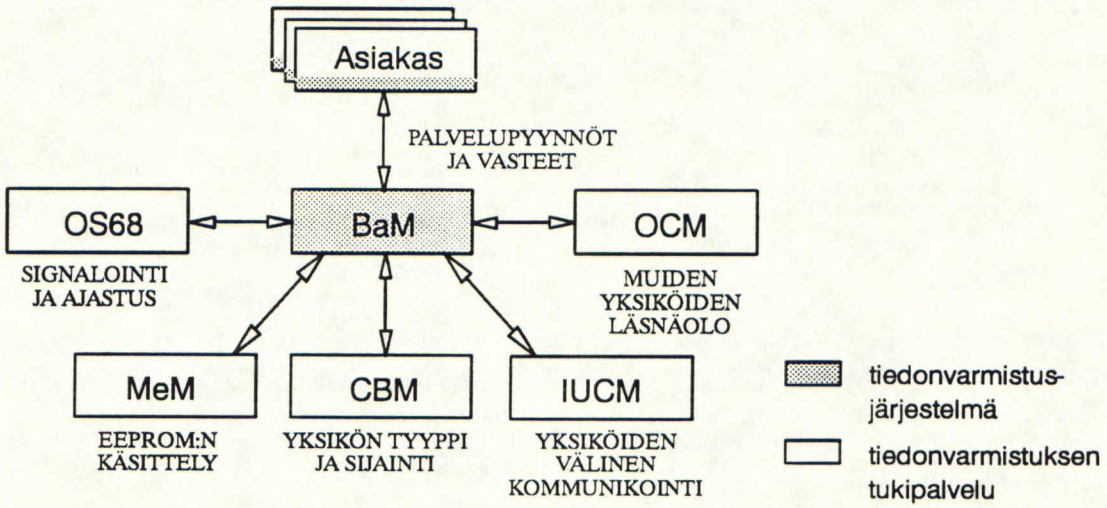
Sääntö 2.2. ei ole yhtään vielä vapaata alempaan ryhmään kuuluvaa yksikköä, muodostetaan epäsäännöllinen tähti keskusyksikkönään mainittu yksinäinen yksikkö.

5 KÄYTÄNNÖN JÄRJESTELMÄ

5.1 Toteutusympäristö

Ohjelmisto toteutetaan kokonaan C++ -kielellä. Suunnittelussa on pyritty oliopohjaiseen lähestymistapaan ottaen huomioon reaaliaikajärjestelmän erityispiirteet. Koodi on käännetty Motorola MC68302-mikroprosessoria varten Intermetricsin kääntäjällä. C++ -toteutuksen etuna on helposti päivitettävä modulaarinen rakenne, jonka osia voidaan käyttää muissa yhteyksissä uudelleen. Huonona puolena on ohjausvuon ajoittainen katoaminen oliorajapintoja ylitettäessä.

Toimintansa tueksi tiedonvarmistusjärjestelmä tarvitsee joitakin muita ohjelmistokomponentteja ja käyttöjärjestelmän (kuva 13); MeM hoitaa EEPROM:n käsittelyn matalalla tasolla, CBM (*Control Block Manager*) lukee raudasta pistoyksikön sijainnin ja tyypin, OCM:a (*Object Configuration Manager*) käytetään seuraamaan yksittäisten pistoyksiköiden läsnäoloa solmussa ja IUCM mahdollistaa sanomanvaihdon pistoyksiköiden kesken.



Kuva 13 Tiedonvarmistusjärjestelmää ympäröivät ohjelmistokomponentit

Käyttöjärjestelmänä on OS68 /16/, jonka palveluja tarvitaan ohjelmaprosessien luomiseen, prosessien väliseen signalointiin ja ajastuspalveluihin.

5.2 Prosessijako käyttöjärjestelmätasolla

Edellä esitettyjen vaatimusten pohjalta on päädytty seuraavanlaiseen ohjelmiston prosessijakoon:

Tiedonvarmistusjärjestelmän palvelut saadaan rajapintaluokan kautta. Koska palvelun tuotttajan ja palveltavan asiakkaan dynamiikka on erilainen, kukin asiakas ottaa rajapintaluokan instanssin johonkin omaan prosessiinsa.

Järjestelmän rajapintaluokka keskustelee pistoyksikkökohtaisen paikallisen tiedonvarmistuspalvelimen kanssa.

Paikallinen palvelin saattaa joutua viettämään paljon aikaa käsitellen pistoyksikön EEPROM:a, joten pistoyksiköiden välinen solmutason yhteistoiminta on irrotettu paikallisesta palvelimesta erilliseen solmutason palvelinprosessiin, joka ylläpitää omaa tietämystä solmutason toimintaan vaikuttavista tiedostotapahtumista.

Solmutason palvelimen toimintaa voidaan helpottaa erottamalla pistoyksiköiden välinen sanomanvaihto erillisen sanoma–ajurin tehtäväksi. Tällöin kaikki solmutason palvelimen tarvitsema sanomanvaihto voidaan järjestää käyttöjärjestelmäsignaaleilla ilman IUCM:n ja käyttöjärjestelmäsignaalien rinnakkaista käyttöä. Ajuri piilottaa sisäänsä pistoyksiköiden välisen sanomaprotokollan ja sanomien reitityksen sekä tiedonvarmistusjärjestelmän solmutason käynnistymistä tehostavan ajastetun sanomanlähetysmahdollisuuden. Myöhemmässä vaiheessa saattaa tulla ajankohtaiseksi sanomanvaihdon nopeuttaminen ottamalla käyttöön käyttöjärjestelmäsanomien linkkikäsittelijä; sanomanvaihtopalvelun kätkevän ajurin käyttäminen helpottaa tätä mahdollista muutosta.

Järjestelmälle on määritelty tiedostojen varmistusarkkitehtuurin määrittelevä pistoyksikkökohtainen päätöksentekoelin, joka seuraa solmussa tapahtuvia pistoyksiköiden saapumisia ja poistumisia ja reagoida niihin päättämällä mahdollisesta levityssuunnitelman muutoksesta ja käskyttämällä muutokset oman pistoyksikön solmutason palvelimelle. Itsenäisen toiminnan mahdollistamiseksi päätöksentekoelin (topologi) on sijoitettu omaan prosessiinsa.

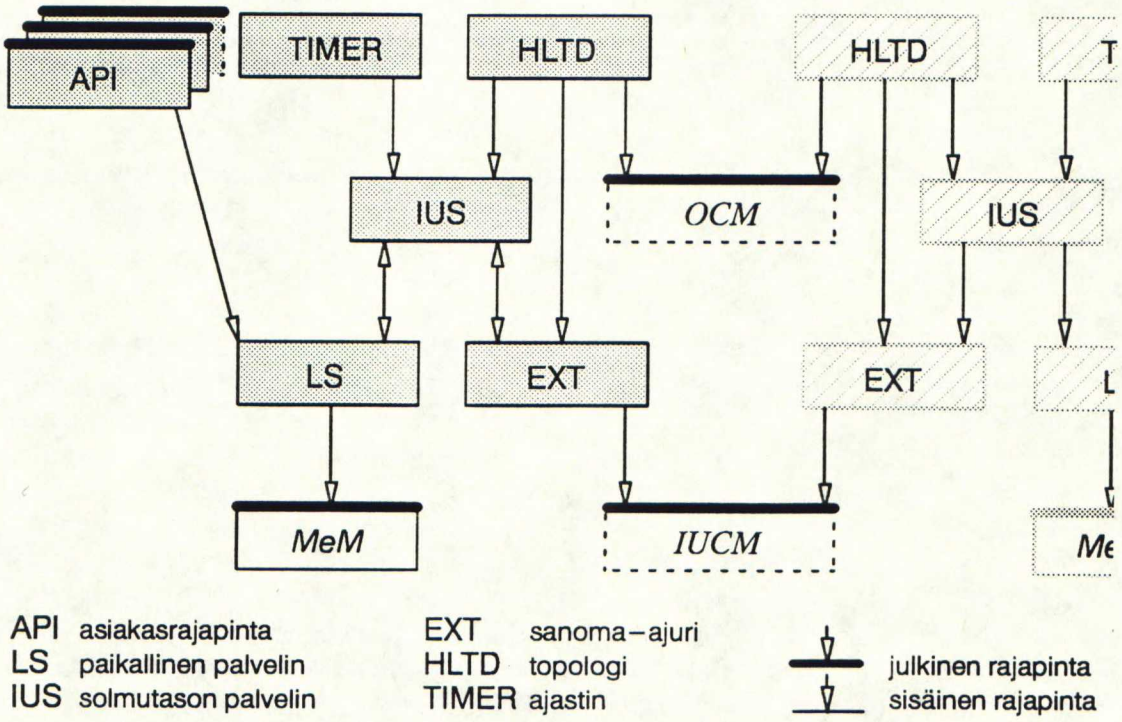
Tietyissä tapauksissa on mahdollista, että pistoyksikön toiselle yksikölle tekemät tiedostokopiot katoavat (kohdeyksikkö esimerkiksi poistetaan solmusta tai

EEPROM ajan myötä korruptoituu). Aiheutuva haitta voidaan välttää tarkistamalla ajoittain, että kaikista levitettävistä tiedostoista on solmussa olemassa kopio. Toiminnasta vastaava täydennyslevitysajastin on toteutettu erillisenä prosessina.

Yhteenvetona voidaan koota järjestelmän prosessit:

1. Asiakkaan prosessi
2. Paikallinen palvelin
3. Solmutason palvelin
4. Pistoyksiköiden välisen sanomaliikenteen ajuri
5. Päätöksentekoelin (topologi)
6. Täydennyslevitysajastin

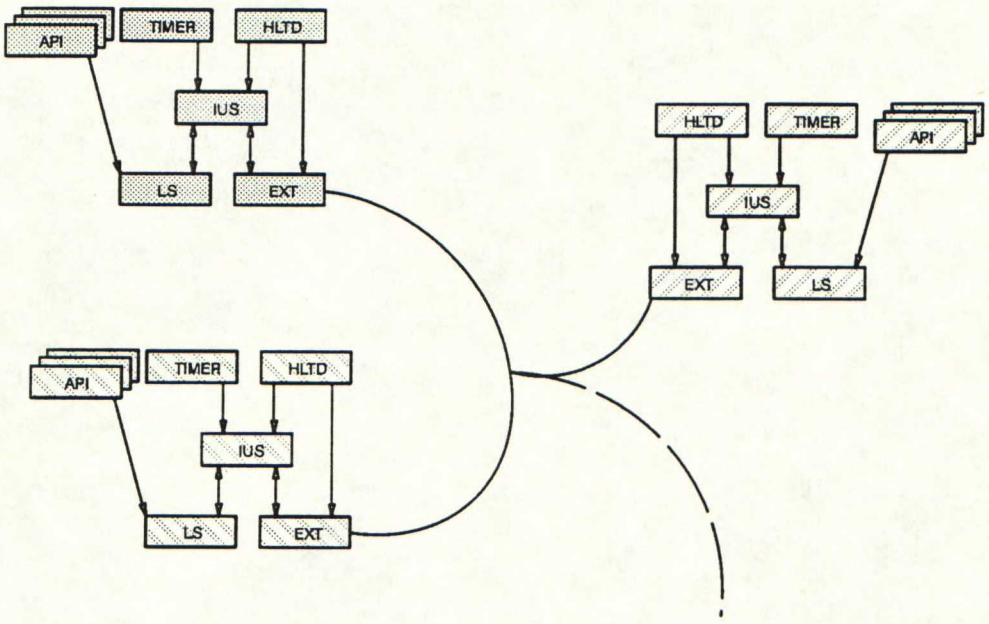
Kuvassa 14 esitetään järjestelmän prosessien keskinäiset sidokset siten, että käskytettävä prosessi sijaitsee nuolen kärjen puolella. Harmaat ja vinoviivoitetut laatikot kuvaavat tiedonvarmistusjärjestelmän omia prosesseja; tukikomponentit on esitetty valkoisina ja pistoyksiköiden rajalla katkoviivoitettuina. Julkiset rajapinnat on esitetty vahvennettuina vaakaviivoina. Kukin paikallinen palvelin palvelee vaihtelevaa määrää asiakkaita.



Kuva 14 Järjestelmän prosessien keskinäiset sidokset ja ulkoiset tukikomponentit

Tiedonvarmistusjärjestelmä on suunniteltu siten, että sen prosessien keskinäiset suoritusprioriteetit ovat samat; tällä ratkaisulla pyritään välttämään ongelmia, joita voi syntyä muutettaessa asiakasohjelmistokomponenttien prioriteetteja. Asiakkailla on pääsääntöisesti sama tai korkeampi prioriteetti kuin solmua palvelevalla tiedonvarmistuksella.

Päätöksentekoelin seuraa solmun konfiguraatiota OCM:n kautta ja välittää tiedot muutoksista sanoma–ajurille, joka siten on koko ajan tietoinen ympäristöstään eikä kuormita solmua turhilla sanomanlähetysyrityksillä. Järjestelmään kuuluvat pistoyksiköt keskustelevat keskenään sanoma–ajurien (kuva 15) ja pistoyksiköiden sanoma–ajurit IUCM:n kautta. Kuvan mukaisesti solmutason palvelinten ei tarvitse seurata muiden pistoyksiköiden läsnäoloa, vaikka ne toimivat myös väliportaina paikallisten palvelinten ja ulkoisten pistoyksiköiden välillä.



Kuva 15 Pistoyksiköiden tiedonvarmistusprosessien väliset suhteet

Ajurissa on lisäpiirteenä mahdollisuus ajastettuun lähetykseen, jolloin solmutason palvelin määrää lähtevälle sanomalle ajan, jonka kuluessa paikalle ilmaantuva vieras pistoyksikkö saa sanoman. Näin voidaan toteuttaa ryhmälähetykset kaikille solmun pistoyksiköille siten, että ne esimerkiksi hieman eriaikaisesti käynnistyessään kuitenkin saavat kukin sanoman; muutoin olisi esimerkiksi solmun päällekytkemi-

sen jälkeen suoritettava kaikkien pistoyksiköiden kesken aikavalvottu esittäytymiskättely, joka hidastaa käynnistymistä.

5.3 Prosessien tilakäyttäytyminen ja keskinäinen signalointi

Järjestelmän prosessit reagoivat ulkoisiin herätteisiin, joita ovat prosessin sanomajonoon saapuvat signaalit. Signaaleja lähettävät asiakkaan lisäksi kaikki järjestelmän prosessit tiettyjen sovittujen sääntöjen mukaisesti. Ilman sovellusrajapinnasta asiakkaalta tulevia herätteitä tiedonvarmistusjärjestelmä nukahtaa kaikki tarpeelliset tehtävät suoritettuaan odottamaan uusia palvelupyyntöjä. Järjestelmän herättää ajoittain myös täydennyslevitysajastin, joka antaa herätteen solmutason palvelimelle.

Seuraavassa on ensin lyhyesti esitelty juoksevasti numeroituina kaikki järjestelmässä määritellyt signaalit ja sen jälkeen esitetään sallittujen signaalien kulku ristiin taulukoituna siten, että kunkin prosessin osalta luetellaan sen herätteeksi hyväksymät ja lähettämät signaalit. Signaalimäärittelyn yhteydessä on myös kuvattu signaalin sallittu kulku prosessilta toiselle. Mikäli prosessi vastaanottaa signaalin jota ei ole määritelty hyväksytyksi, on kyse virhetilanteesta, josta toivutaan parhaiten poistamalla signaali prosessin signaalijonosta.

5.3.1 Signaalimäärittelyt

1 *Palvelupyyntö: 'Luo paikallinen tiedosto'*
2 *Palvelupyyntö: 'Varmista paikalliseen tiedostoon'*
3 *Palvelupyyntö: 'Palauta paikallinen tiedosto'*
4 *Palvelupyyntö: 'Hävitä paikallinen tiedosto'*
5 *Palvelupyyntö: 'Luo levitettävä tiedosto'*
6 *Palvelupyyntö: 'Varmista levitettävään tiedostoon'*
7 *Palvelupyyntö: 'Palauta levitettävästä tiedostosta'*
8 *Palvelupyyntö: 'Hävitä levitettävä tiedosto'*

Edellä mainitut ovat kaikki sovellusrajapinnasta tulevia palvelupyyntöjä.

Osapuolet:

API⇒LS

9 *Palvelupyyntö: 'Nouda tiedoston varjokopio'*

Sekä sovellusrajapinnasta tuleva palvelupyyntö että järjestelmän sisäinen signaali.

Osapuolet:

LS⇒IUS⇒SANOMA-AJURI⇒SANOMA-AJURI⇒IUS

10 *Vastaus palvelupyyntöön*

Sisältää tilatietoja ja mahdollisen vastedatan.

Osapuolet:

LS⇒API

11 *Rekisteröityminen OCM:lle*

Ilmaisee, että olio haluaa rekisteröityä OCM:n kautta koko solmun seurattavaksi. Signaali sisältää pistoyksikön tiedonvarmistusryhmätunnuksen.

Osapuolet:

HLTD⇒OCM

12 *Rekisteröitymisen peruutus OCM:lle*

Ilmaisee olion haluavan poistua solmutason toiminnasta (OCM ilmoittaa tapahtumasta muille pistoyksiköille).

Osapuolet:

HLTD⇒OCM

13 *Oliotapahtuma*

Ilmoitus tietyn olion solmutason poistumisesta tai saapumisesta.

Osapuolet:

OCM⇒HLTD

14 *Pistoyksikkötapahtuma*

Ilmoitus tietyn pistoyksikön poistumisesta tai saapumisesta.

Osapuolet:

HLTD⇒SANOMA-AJURI

15 *Valmiusilmoitus*

Osapuolet:

IUS⇒HLTD

16 *Spontaani identiteetti-ilmoitus*

Spontaanisti lähetettävä signaali, joka sisältää pistoyksikön tiedonvarmistuksellisen identiteettitunnisteen.

Osapuolet:

IUS⇒SANOMA-AJURI⇒SANOMA-AJURI⇒IUS

17 *Vastaus identiteetti–ilmoitukseen*

Vastaussignaali, joka sisältää vastaavan pistoyksikön tiedonvarmistuksellisen tunnisteen.

Osapuolet:

IUS⇒SANOMA–AJURI⇒SANOMA–AJURI⇒IUS

18 *Lupa solmutason toimintaan*

Signaali, joka ilmaisee pistoyksikön omien tiedostojen olevan ajantasalla ja että paikallinen palvelin saa aloittaa normaalin toiminnan. Sisältää solmutunnisteen.

Osapuolet:

IUS⇒LS

19 *Lista sallituista varmistuslinkeistä*

Osapuolet:

HLTD⇒IUS

20 *Ilmoitus muuttuneesta varmistustopologiasta*

Osapuolet:

HLTD⇒AJASTIN

21 *Irtikytkeytymiskäsky*

Käsky irtautua kaikesta pistoyksiköiden välisestä toiminnasta. Käytetään vakavissa vikatilanteissa.

Osapuolet:

LS⇒HLTD

LS⇒SANOMA–AJURI

LS⇒IUS

22 *Pyyntö listata tiedostot*

Osapuolet:

IUS⇒LS

23 *Lista tiedostoista*

Sisältää täydelliset tiedostotunnisteet, tiedostojen pituudet, versiotunnisteet ja alkuperäisen omistajan.

Osapuolet:

LS⇒IUS

IUS⇒SANOMA–AJURI⇒SANOMA–AJURI⇒IUS

24 *Ilmoitus tiedostoversiosta*

Sisältää täydellisen tiedostotunnisteen, tiedoston pituun, versiotunnisteen ja tiedon signaalin lähettäjästä.

Osapuolet:

IUS⇒SANOMA–AJURI⇒SANOMA–AJURI⇒IUS

25 *Ulkoinen hävityspyyntö*

Käsky hävittää määritelty tiedosto ja kaikki sen kopiot.

Osapuolet:

IUS⇒LS

IUS⇒SANOMA–AJURI⇒SANOMA–AJURI⇒IUS⇒LS

LS⇒IUS⇒SANOMA–AJURI⇒SANOMA–AJURI⇒IUS⇒LS

26 *Ulkoinen talletuspyyntö*

Sisältää tiedoston ja sen tunnisteet (täydellinen nimi, pituus ja versio).

Osapuolet:

IUS⇒SANOMA–AJURI⇒SANOMA–AJURI⇒IUS⇒LS

LS⇒IUS⇒SANOMA–AJURI⇒SANOMA–AJURI⇒IUS⇒LS

27 *Vastaus ulkoiseen talletuspyyntöön*

Ilmoitus tallennuksen onnistumisesta (vastaus signaaliin 26).

Osapuolet:

LS⇒IUS

LS⇒IUS⇒SANOMA–AJURI⇒SANOMA–AJURI⇒IUS

28 *Ulkoinen palautuspyyntö*

Pyyntö toimittaa signaalin lähettäjälle kopio signaalissa määritellystä tiedostosta.

Osapuolet:

IUS⇒SANOMA–AJURI⇒SANOMA–AJURI⇒IUS⇒LS

29 *Vastaus ulkoiseen palautuspyyntöön*

Sisältää tiedon palautuksen onnistumisesta sekä tiedoston tunnisteineen.

Osapuolet:

LS⇒IUS⇒SANOMA–AJURI⇒SANOMA–AJURI⇒IUS

30 *Ilmoitus spontaanista tiedoston hävittämisestä*

Signaali sisältää tiedoston täydelliset tunnistetiedot. Käytetään ilmaisemaan, että jokin paikallinen palvelin on tilanahtauden vuoksi hävittänyt

vieraan tiedoston, tai että oma solmutason palvelin on havainnut ettei tiedostosta ole solmussa yhtään kopiota.

Osapuolet:

IUS⇒LS

LS⇒IUS⇒SANOMA-AJURI⇒SANOMA-AJURI⇒IUS⇒LS

31 *Vastaus varjokopion noutopyyntöön*

Sisältää tiedon pnoudon onnistumisesta sekä tiedoston tunnisteineen.

Osapuolet:

IUS⇒LS

32 *Tarkistuspyyntö*

Signaali ilmaisee, että on syytä tarkistaa tiedostojen levityksen kattavuus.

Osapuolet:

AJASTIN⇒IUS

33 *Pistoyksiköiden välinen sanoma*

Sanoma-ajurien välisen keskinäisen protokollan mukainen signaali; sisältönä voi olla mikä tahansa solmutason palvelimien ymmärtämä sanoma.

Osapuolet:

IUS⇒SANOMA-AJURI⇒SANOMA-AJURI⇒IUS

34 *Pistoyksiköiden välinen sanomankuittaus*

Ilmaisee tässä signaalissa määritellyn signaalin saapuneen määritellylle pistoyksikölle.

Osapuolet:

SANOMA-AJURI⇒SANOMA-AJURI

35 *Pistoyksiköiden välinen sanomanhylkäys*

Solmutason toiminnasta irti kytketty sanoma-ajuri ilmaisee tällä signaalilla, että toiselta sanoma-ajurilta vastaanotettu sanoma on vastaanotettu, mutta jätetty käsittelemättä. Tämä signaali tarvitaan, ettei vastapää turhaan yritä sanoman uudelleenlähetystä.

Osapuolet:

SANOMA-AJURI⇒SANOMA-AJURI

5.3.2 Signalointitaulukot

Järjestelmän osista lähtevät signaalit:

Asiakasrajapinnalta

Paikalliselle palvelimelle—: 1, 2, 3, 4, 5, 6, 7, 8

Paikalliselta palvelimelta

Asiakasrajapinnalle ——: 10

Solmutason palvelimelle—: 21, 23, 25, 26, 27, 29, 30

Topologille————: 21

Sanoma–ajurille————: 21

Solmutason palvelimelta

Paikalliselle palvelimelle—: 18, 22, 25, 26, 28, 30

Topologille————: 15

Sanoma–ajurille————: 16, 17, 23, 24, 25, 26, 27, 28, 29, 30

OCM:lta

Topologille————: 13

Topologilta

Solmutason palvelimelle—: 19

OCM:lle————: 11, 12

Sanoma–ajurille————: 14

Ajastimelle————: 20

Sanoma–ajurilta

Solmutason palvelimelle—: 16, 17, 23, 24, 25, 26, 27, 28, 29, 30

Sanoma–ajurille————: 33, 34, 35

Ajastimelta

Solmutason palvelimelle—: 32

Järjestelmän osiin saapuvat signaalit:

Asiakasrajapinnalle

Paikalliselta palvelimelta—: 10

Paikalliselle palvelimelle

Asiakasrajapinnalta———: 1, 2, 3, 4, 5, 6, 7, 8

Solmutason palvelimelta—: 18, 22, 25, 26, 28, 30

Solmutason palvelimelle

Paikalliselta palvelimelta—: 21, 23, 25, 26, 27, 29, 30

Topologilta————————: 19

Sanoma–ajurilta————: 16, 17, 23, 24, 25, 26, 27, 28, 29, 30

Ajastimelta————————: 32

OCM:lle

Topologilta————————: 11, 12

Topologille

Paikalliselta palvelimelta—: 21

Solmutason palvelimelta—: 15

OCM:lta————————: 13

Sanoma–ajurille

Paikalliselta palvelimelta—: 21

Solmutason palvelimelta—: 16, 17, 23, 24, 25, 26, 27, 28, 29, 30

Topologilta————————: 14

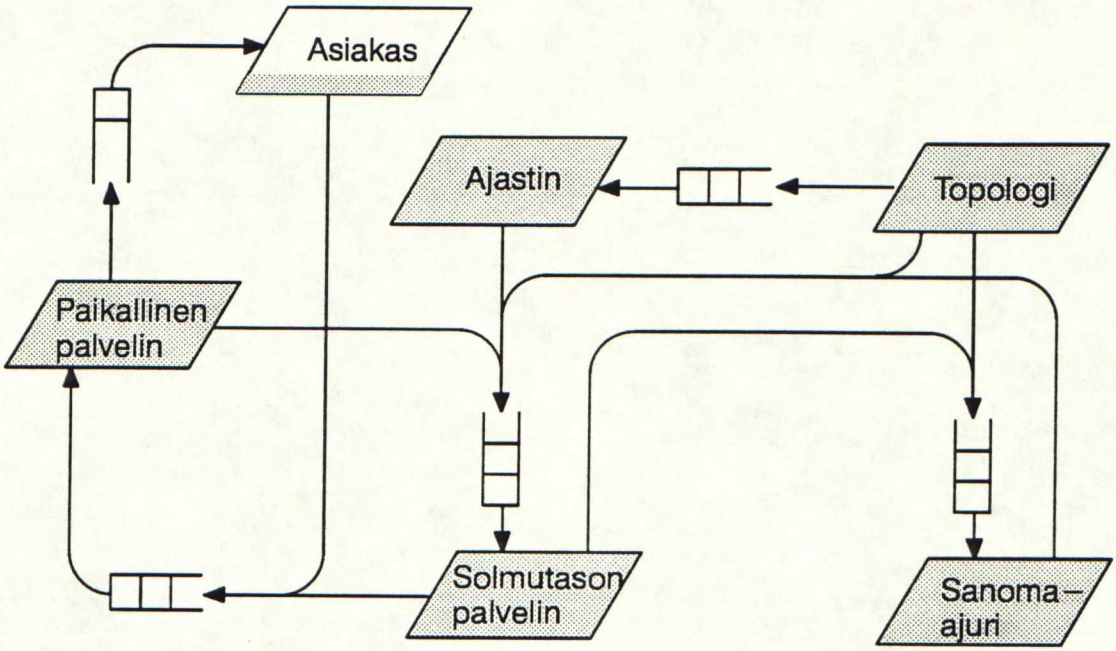
Sanoma–ajurilta————: 33, 34, 35

Ajastimelle

Topologilta————————: 20

Kuvasta 16 ilmenevät pistoyksiköllä sijaitsevien tiedonvarmistusprosessien väliset signalointisidokset DARTS–esitystavan mukaisena yhteenvetona /17/; selkeyden vuoksi paikalliselta palvelimelta topologille ja sanoma–ajurille mahdollisesti läh-

tevien irtikytkeytymiskäskyjen reitti on jätetty piirtämättä. Eri pistoyksiköiden sanoma–ajurit kommunikoivat lisäksi keskenään kaksisuuntaisesti. Kaikki sidokset lukuunottamatta asiakkaan ja paikallisen palvelimen välistä liikennöintiä ovat lukkotilanteiden välttämiseksi löyhiä eli signaalin lähettänyt prosessi ei jää odottamaan pelkkää vastesignaalia, vaan hyväksyy ja prosessoi myös muut sanomat.



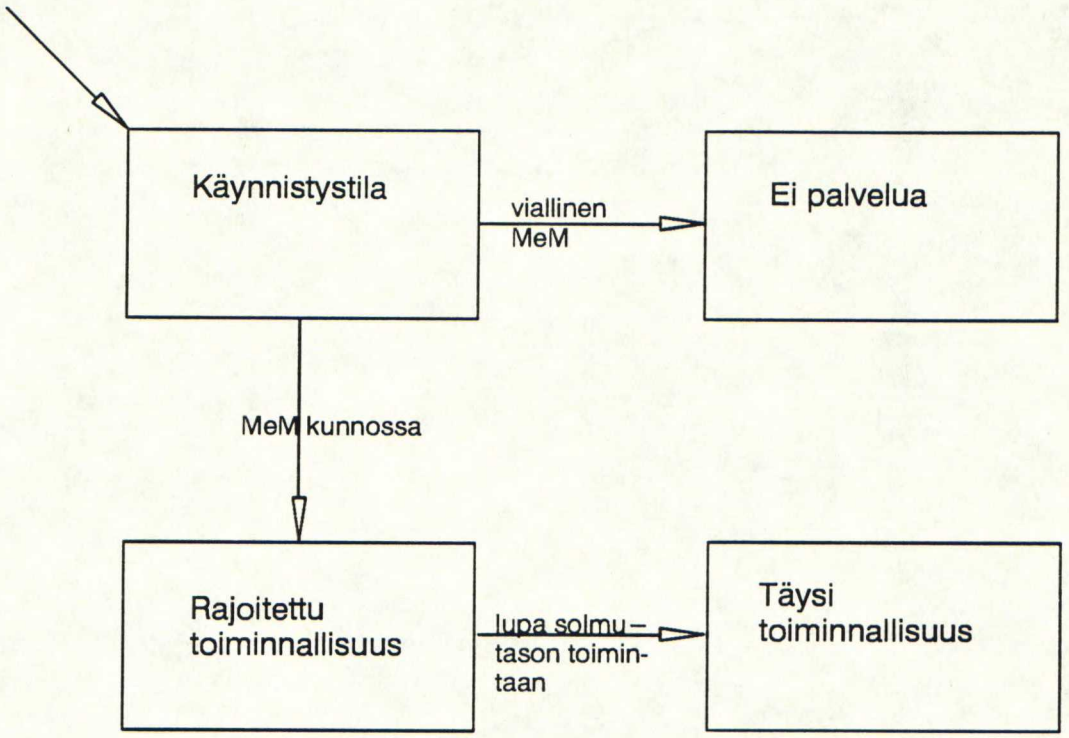
Kuva 16 Järjestelmän prosessien väliset signalointisidokset

Pistoyksiköiden välisessä sanomanvaihdossa on varauduttava siihen, että lähetetty signaali syystä tai toisesta katoaa matkan varrella, joten solmutason palvelin ei milloinkaan saa jäädä ikuisesti odottamaan mahdollisesti kaipaamaansa vastetta, vaan toimintaa on jatkettava viimeistään erikseen asetetun aikavalvonnan lauettua. Tästä syystä erityisesti pistoyksiköiden välisen signaloinnin on oltava löyhästi kytkettyä.

Seuraavassa esitellään kullekin prosessille löytyneet tilat; käytännössä tilakoneajatus toteutuu siinä toteutusluokassa, jonka tehtäväksi on annettu ylläpitää kyseisen prosessin toimintoja. Prosessien aikatason toiminta jäsentyy kappaleessa 5.4.2, jossa käsitellään järjestelmän oliokommunikaatiomallia.

5.3.3 Paikallinen palvelin

Paikallinen palvelin käynnistää käyttöjännitteen päälle kytkemisen jälkeen MeM:n, joka toimii EEPROM–ajurina. Mikäli ajuri ei käynnisty esimerkiksi vioittuneen tai poistetun muistipiirin takia, paikallinen palvelin siirtyy tilaan, jossa se ei tarjoa muille järjestelmän prosesseille mitään palveluja (kuva 17). Tällöin käsketään muiden varmistusjärjestelmän osien lopettaa toimintansa ja asiakkailta tuleviin palvelupyyntöihin vastataan, ettei palvelu ole käytössä. EEPROM:n käyttökelpoisuuspäätös on luonteeltaan lopullinen, joten tästä tilasta ei voida siirtyä pois milloinkaan.



Kuva 17 Paikallisen palvelimen sisäisen toiminnan päätilat

Järjestys, jossa muiden järjestelmän osien mahdollisesti käsketään lopettaa toimintansa, on merkitsevä, jotta erityisesti pistoyksiköiden väliset herätevirrat saadaan vaimennettua mahdollisimman nopeasti: ensin käskytetään topologi, koska tällöin kaikki solmun muut pistoyksiköt saavat mahdollisimman nopeasti tiedon tämän pistoyksikön ongelmista eivätkä ne enää turhaan yritä lähettää sanomia. Seuraavaksi kytketään pistoyksiköiden välisen liikenteen hoitava sanoma–ajuri tilaan, jossa se lakkaa reitittämästä muilta yksiköiltä tulevia sanomia oman yksikön varmistusjär-

jestelmän prosesseille ja absorboi mahdollisesti ulos lähdössä olevat omat sanomat, ja lopuksi käsketään solmutason palvelimen hiljentyä.

Mikäli EEPROM–ajuri käynnistyy normaalisti, paikallinen palvelin aloittaa rajoitetun toiminnan vastaamalla asiakkaiden esittämiin paikallisia tiedostoja koskeviin palvelupyyntöihin; levitettäviksi tarkoitettuja tiedostoja ei vielä voida näyttää asiakkaille, sillä tiedostot eivät välttämättä ole solmutasolla ajan tasalla. Ajantasaistus kuuluu solmutason palvelimen tehtäviin, joten paikallisen palvelimen on tässä tilassa reagoitava myös kaikkiin solmutason palvelimelta tuleviin herätteisiin. Tiedostojen ollessa ajan tasalla, solmutason palvelin antaa paikalliselle palvelimelle luvan myös levitettäviksi tarkoitettuihin tiedostoihin viittaavien palvelupyyntöjen täyttämiseen, jolloin signaalijonosta otetaan vuorollaan käsittelyyn kaikki signaalit.

Mikäli tässä vaiheessa todetaan vikoja EEPROM:ssa, ne pyritään peittämään asiakkailta eikä palvelimen sisäinen tila voi tässäkään tapauksessa enää muuttua.

Reaktiot ulkoisiin herätteisiin:

Palvelupyyntö: 'Luo paikallinen tiedosto'
Palvelupyyntö: 'Varmista paikalliseen tiedostoon'
Palvelupyyntö: 'Palauta paikallinen tiedosto'
Palvelupyyntö: 'Hävitä paikallinen tiedosto'

> Nämä herätteet hyväksytään koska tahansa käynnistymisen jälkeen, mikäli EEPROM–ajuri on käynnistynyt ja käyttökelpoinen. Herätteet aiheuttavat MeM:n käskyttämisen ja vasteen asiakkaalle.
> Luonti– ja palautuspyynnöillä on suorituskykysyistä etuoikeus ennen EEPROM:iin kirjoittamista aiheuttavia varmistus– ja hävityspyyntöjä. Vapaan muistin käydessä vähiin saattaa luontipyyntö aiheuttaa vieraiden tiedostojen hävittämisiä. Tällöin solmutason palvelimelle lähetetään tiedostoja vastaavat *Ilmoitus spontaanista tiedoston hävittämisestä*–signaalit.

Palvelupyyntö: 'Luo levitettävä tiedosto'
Palvelupyyntö: 'Varmista levitettävään tiedostoon'
Palvelupyyntö: 'Palauta levitettävästä tiedostosta'
Palvelupyyntö: 'Hävitä levitettävä tiedosto'

> Nämä herätteet hyväksytään vasta solmutason toimintaluvan saamisen jälkeen. Herätteet aiheuttavat MeM:n käskyttämisen ja vasteen asiakkaalle.
> Hävityspyyntö aiheuttaa vastaavan *Ulkoinen hävityspyyntö*–signaalin lähettämisen solmutason palvelimelle.

Palvelupyyntö: 'Nouda tiedoston varjokopio'

> Käsitellään aina, jos EEPROM–ajuri on käynnissä. Signaali välitetään edelleen solmutason palvelimelle ja jäädään odottamaan vastausta, joka

on *Vastaus varjokopion noutopyyntöön*. Kun vastaus on saatu, verrataan tulosta mahdollisesti omalta yksiköltä löytyneeseen versioon ja annetaan vaste asiakkaalle.

Pyyntö listata tiedostot
Käsitellään aina, jos EEPROM–ajuri on käynnissä. Vastauksena lähetetään tiedostoluettelo.

Ulkoinen hävityspyyntö
Käsitellään aina, jos EEPROM–ajuri on käynnissä. Tähän herätteeseen ei vastata, vaan ainoastaan käskytetään MeM.

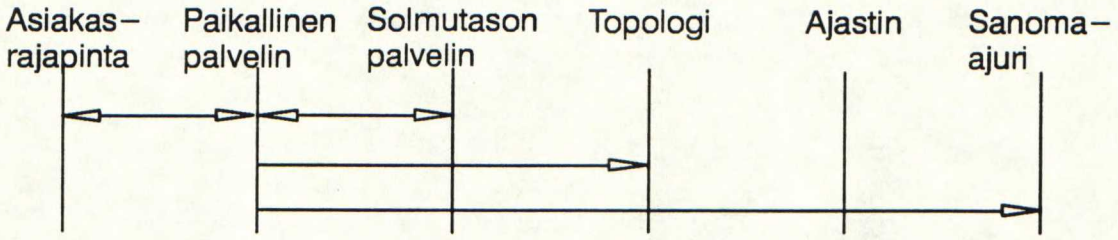
Ulkoinen talletuspyyntö
Käsitellään aina, jos EEPROM–ajuri on käynnissä. Vastauksena lähetetään *Vastaus ulkoiseen talletuspyyntöön*–signaali.

Lupa solmutason toimintaan
Käsitellään aina, jos EEPROM–ajuri on käynnissä. Palvelimen tila muuttuu ja herätteen mukana saatu solmutunniste otetaan käyttöön välittömästi.

Ilmoitus spontaanista tiedoston hävittämisestä
Käsitellään aina, jos EEPROM–ajuri on käynnissä. Koska tiedostosta tehty ulkoinen kopio on hävinnyt, lähetetään solmutason palvelimelle *Ulkoinen talletuspyyntö*.

Ulkoinen palautuspyyntö
Käsitellään aina, jos EEPROM–ajuri on käynnissä. Aiheuttaa MeM:n käskyttämisen ja vasteen palauttamisen signaalissa *Vastaus ulkoiseen palautuspyyntöön*.

Kuvasta 18 ilmenevät graafisena yhteenvetona paikallisen palvelimen signalointisidokset muihin tiedonvarmistusjärjestelmän prosesseihin; nuolen kärki osoittaa signaalin vastaanottajaa kohti.



Kuva 18 Paikallisen palvelimen signalointisidokset muihin järjestelmän prosesseihin

5.3.4 Solmutason palvelin

Solmutason palvelin on laaja kokonaisuus, joka käynnistyksen yhteydessä vastaa ensin pistoyksikön ja sitten solmun tiedonvarmistustunnisteen päättämisestä, levitettäviksi määrättyjen tiedostojen ajantasaistamisesta solmutasolla ja normaaliin toimintatilaan päästyään oman pistoyksikön tiedostojen levittämisestä ja vanhentuneiden kopioiden hävittämisestä paikallisen palvelimen antamien herätteiden mukaisesti. Vastaavasti palvelin tukee muiden pistoyksiköiden palvelimia niiden suorittaessa tehtäviään solmutasolla.

Kaikki kommunikointi muiden pistoyksiköiden palvelinten kanssa tapahtuu sanoma–ajurin kautta eli normaalein käyttöjärjestelmäsignaalein.

Reaktiot ulkoisiin herätteisiin:

Lista sallituista varmistuslinkeistä

Lista talletetaan vastaisia levityspyyntöjä varten. Mikäli kysessä on ensimmäinen vastaanotettu luettelo, paikalliselle palvelimelle lähetetään signaali *Lupa solmutason toimintaan* merkiksi siitä että paikalliset tiedostot on ajantasaistettu ja että solmutason palvelin on valmis käsittelemään mahdollisia tiedostonlevityspyyntöjä.

Lista tiedostoista

Toisen pistoyksikön lähettämänä tämä heräte tarkoittaa, että kysyjä haluaa selvittää omista tiedostoistaan olemassa olevien kopioiden sijainnin ja ajantasaisuuden, joten luettelon tiedostoja verrataan omiin tiedostoihin; mikäli omalta yksiköltä löytyy tiedostosta uudempi tai sama versio, kysyjälle lähetetään *Ilmoitus tiedostoversiosta* ja jos oma tiedosto on vanhempi, paikallista palvelinta pyydetään omistajaa vaivaamatta hävittämään se. Mikäli oman yksikön tiedostoista ei vielä ole tietoa, pyydetään luettelo omalta paikalliselta palvelimelta (*Pyyntö listata tiedostot* ja vasteena *Lista tiedostoista*).

Ulkoinen hävityspyyntö

Jos heräte on lähtöisin omalta yksiköltä, se lähetetään kaikille muille yksiköille hyödyntäen sanoma–ajurin ajastettua lähetysoptiota.
Toiselta yksiköltä tullut heräte välitetään paikalliselle palvelimelle.

Ulkoinen talletuspyyntö

Toiselta pistoyksiköltä lähtöisin oleva heräte välitetään paikalliselle palvelimelle, jolta odotetaan *Vastaus ulkoiseen talletuspyyntöön*. Tämä vastaus välitetään alkuperäisen herätteen lähettäjälle.
Omalta yksiköltä tulleen herätteen saatuaan palvelimen tulee päätää, mille ulkoiselle yksikölle tiedosto kopioidaan. Heräte välitetään tälle yksikölle, joka kertoo vastauksessaan (*Vastaus ulkoiseen talletuspyyntöön*), onnistuiko talletus. Epäonnistumisen sattuessa palvelin yrittää talletusta uudelle yksikölle, kunnes yritys on menestyksellinen tai kunnes palvelin

on yrittänyt talletusta kaikille sille sallituille yksiköille. Operaation tulosta ei kerrota paikalliselle palvelimelle.

Ulkoinen palautuspyyntö

Ulkoiselta pistoyksiköltä saatu heräte välitetään paikalliselle palvelimelle, joka vastaa signaalilla *Vastaus ulkoiseen palautuspyyntöön*. Mikäli paikallinen palautus onnistui, vaste välitetään alkuperäiselle herätteen lähettäjälle. Epäonnistuneesta palautuksesta ei lähetetä alkuperäiselle lähettäjälle mitään tietoa, eli kyseessä on täysin löyhä kytkentä. Tämä antaa toisen yksikön solmutason palvelimelle mahdollisuuden jatkaa muiden tehtävien suorittamista heti kun yksittäinen palautuspyyntö on lähetetty; ominaisuus on erittäin tarpeellinen, sillä ulkoista palautuspyyntöä tarvitaan lähinnä solmun käynnistymisen yhteydessä, jolloin nopeus on tärkeä tekijä.

Vastaus ulkoiseen palautuspyyntöön

Tämä heräte ei voi tulla itsenäisenä paikalliselta palvelimelta. Vieraalta pistoyksiköltä vastaanotettuna heräte tulkitaan siten, että solmutason palvelin itse on varhaisemmassa vaiheessa pyytänyt ulkoista palautusta, joten vastaus välitetään paikalliselle palvelimelle herätteenä *Vastaus ulkoiseen talletuspyyntöön*, jos palautus on onnistunut. Tällöin paikallinen palvelin vielä vastaa, mutta vastauksen sisällöllä ei tässä tapauksessa ole merkitystä, sillä mahdollista epäonnistumista ei voi kiertää.

Ilmoitus spontaanista tiedoston hävittämisestä

Omalta yksiköltä tullut ilmoitus välitetään tiedoston alkuperäiselle omistajalle ja vieraalta yksiköltä tullut ilmoitus paikalliselle palvelimelle.

Tarkistuspyyntö

Käynnistyksen aikana tuleva heräte jätetään huomiotta, mutta muulloin tarkistuspyynnön saaminen käynnistää tiedostojen levityksen kattavuuden tarkistuksen:
Paikalliselle palvelimelle lähetetään ensin *Pyyntö listata tiedostot*, johon saadaan vastauksena *Lista tiedostoista*. Luettelosta suodatetaan pois kaikki muiden pistoyksiköiden tiedostot ja jäljelle jäänyt lista lähetetään kaikille läsnä oleville pistoyksiköille (*Lista tiedostoista*). Ulkoiset yksiköt vastaavat tietyn ajan kuluessa tarvittaessa signaaleilla *Ilmoitus tiedostoversiosta*, joiden perusteella päätellään, tarvitseeko omia tiedostoja laittaa täydennyslevitykseen. Mikäli yksittäisestä tiedostosta näyttää olevan olemassa useita kopioita, ylimääräiset poistetaan lähettämällä tarvittava määrä *Ulkoinen hävityspyyntö*-signaaleita. Puuttuvat ulkopuoliset kopiot aiheuttavat *Ilmoitus spontaanista tiedoston hävittämisestä*-signaalien lähettämisen paikalliselle palvelimelle; tämä toimenpide riittää käynnistämään tarpeellisen uudelleenlevityksen.

Spontaani identiteetti-ilmoitus

Vastauksena lähetetään aina *Vastaus identiteetti-ilmoitukseen*.

Vastaus identiteetti-ilmoitukseen

Mikäli heräte saatiin käynnistyksen yhteydessä kun oltiin keräämässä tilastotietoa solmutunnisteen päättämiseksi, päivitetään tietokantaa, muutoin signaali jätetään huomiotta.

Ilmoitus tiedostoversiosta

Jos ilmoitus koskee oman yksikön tiedostoa, versioilmoitusta verrataan oman tiedoston tunnisteisiin (nimi, versio ja pituus). Vertailun tuloksena lähetetään vastaukseksi joko *Ulkoinen hävityspyyntö* tai *Ulkoinen palautuspyyntö*.

Palvelupyyntö: 'Nouda tiedoston varjokopio'

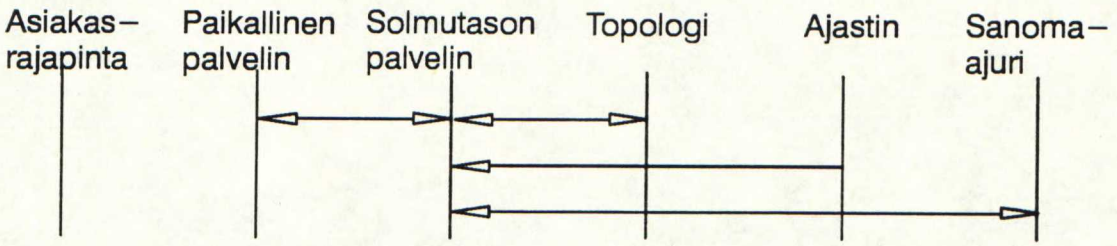
Omalta yksiköltä tuleva heräte välitetään kaikille muille yksiköille ja viimeisimmän version löytämiseksi odotetaan *Ilmoitus tiedostoversiosta* -signaaleja tietyn aikaa. Tämän jälkeen lähetetään mahdollisesti yksi *Ulkoinen palautuspyyntö*. Lopputulos välitetään paikalliselle palvelimelle (*Vastaus varjokopion noutopyyntöön*).

Vieraalta yksiköltä saatuun herätteeseen vastataan signaalilla *Ilmoitus tiedostoversiosta*, jos omalta yksiköltä löytyy mikä tahansa versio tiedostosta.

Irtikytkeytymiskäsky

Signaalijono tyhjennetään täst'edes jatkuvasti ja kaikki vastaanotetut signaalit vapautetaan välittömästi.

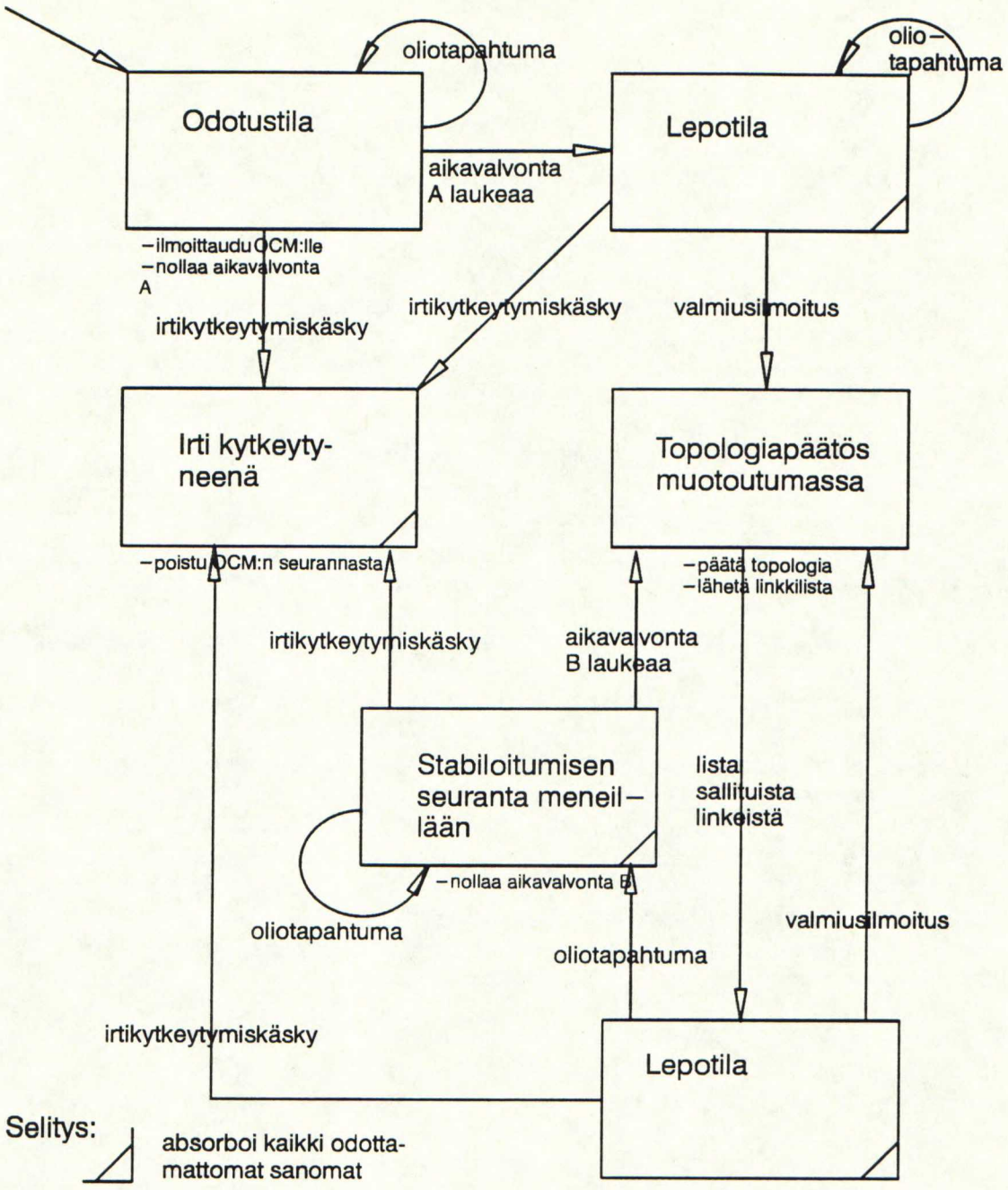
Solmutason palvelimen signalointisidokset muihin järjestelmän prosesseihin ilmenevät kuvasta 19.



Kuva 19 *Solmutason palvelimen ja tiedonvarmistusjärjestelmän muiden prosessien väliset signalointisidokset*

5.3.5 Ympäristömuutosten seuranta ja varmistusarkkitehtuurin suunnittelu

Topologi ilmoittautuu käynnistyessään OCM:lle, jolloin muutkin yksiköt saavat tiedon kyseisen yksikön läsnäolosta, ja seuraa muiden yksiköiden rekisteröitymisiä.



Kuva 20 Topologin sisäisen toiminnan päätilat

Saatuaan valmiusilmoituksen solmutason palvelimelta topologi tekee päätöksen sallituista tiedostonlevityslinkeistä ja antaa sen palvelimelle (kuva 20). Tämän jälkeen topologi uudistaa linkkipäätöksen joka kerta kun solmun pistoyksikkökonfiguraatio muuttuu.

Jotta solmutason palvelimelle ei käskytettäisi uutta linkkilistaa niin nopeasti ettei se ehdi päivitysten ohessa muuta tekemään, topologi suodattaa nopeat vyörynomaiset konfiguraatiomuutokset pois odottamalla tilanteen vakiintumista ennen uutta päätöstä. On mahdollista esimerkiksi, että joku pistoyksikkö on asetettu asennuskehikkoon ikään kuin huulille, jolloin sen sähkönsaanti takalevystä voi räpsyä ja yksikkö nollautuu säännöllisin lyhyin väliajoin ja yksikön tiedonvarmistusjärjestelmä ehtii jokaisen nollautumisen välillä juuri ja juuri ilmoittautua OCM:lle aiheuttaen turhaa kuormitusta muiden yksiköiden tiedonvarmistusjärjestelmille, ellei tilanteen stabiloitumista odoteta ennen johtopäätösten tekoa.

Reaktiot ulkoisiin herätteisiin:

Oliotapahtuma

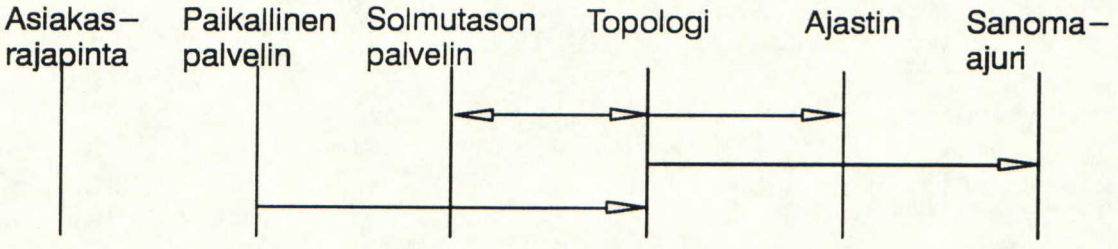
Jonkin pistoyksikön sisältämä tiedonvarmistusolio on joko saapunut solmuun tai poistunut siitä – jos tilanne on stabiili, päätetään uusi topologia päätös ja solmutason palvelimelle lähetetään *Lista sallituista varmistuslinkeistä*. Samalla lähetetään täydennyslevitysajastimelle *Ilmoitus muuttuneesta varmistustopologiasta*.
Sanoma–ajurille lähetetään *Pistoyksikkötapahtuma* aina välittömästi OCM:n ilmoituksen jälkeen, sillä kyseiset signaalit vaikuttavat solmun käynnistyessä siihen kuinka nopeasti sanomat lähtevät solmun muille pistoyksiköille.

Valmiusilmoitus

Topologi tekee välittömästi topologiapäätöksen ja antaa sen paikalliselle palvelimelle signaalina *Lista sallituista varmistuslinkeistä*.

Irtikytkeytymiskäsky

Rekisteröitymisen peruutus OCM:lle lähetetään välittömästi ja samasta hetkestä alkaen kaikki tulevat herätteet jätetään huomiotta ts. kaikki signaalit vapautetaan.

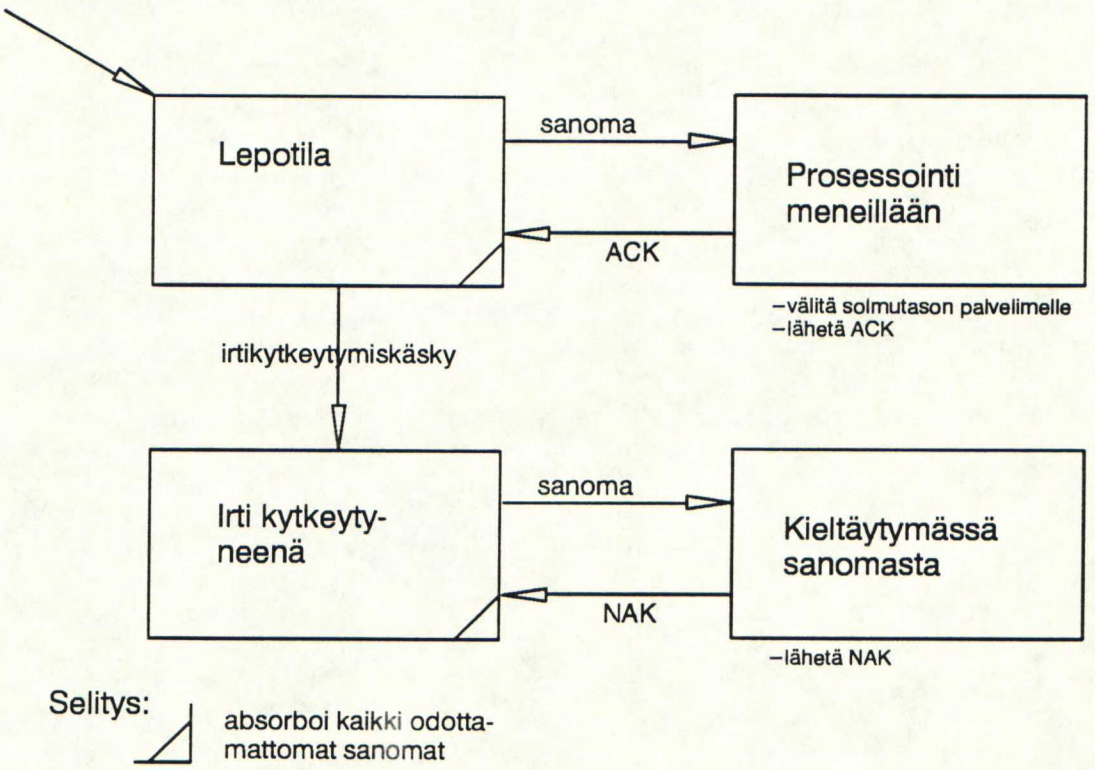


Kuva 21 Signalointisidokset topologin ja järjestelmän muiden prosessien välillä

5.3.6 Pistoyksiköiden väliset sanoma–ajurit

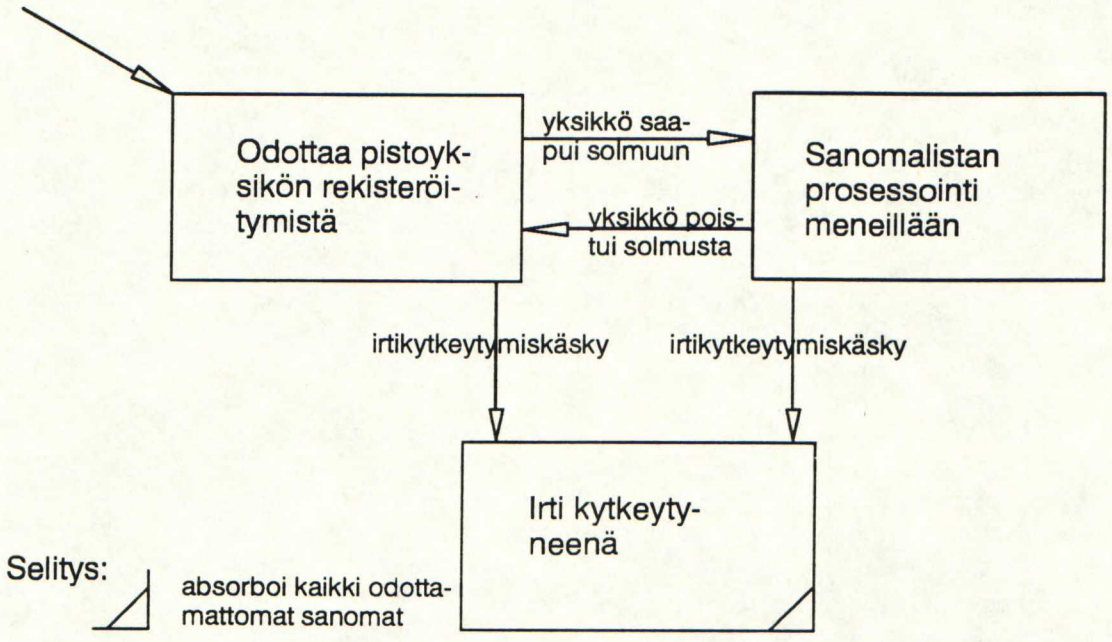
Sanoma–ajurin tilat määräytyvät ulkoisten pistoyksiköiden rekisteröitymisen mukaan erikseen kunkin pistoyksikön osalta, sillä aikavalvotut sanomat on pidettävä muistissa valvonnan laukeamiseen asti ja lähetettävä määrätyille pistoyksiköille aina niiden saapuessa solmuun. Lisäksi sanoma–ajuri voi saada käskyn irtautua solmutason toiminnasta, jolloin sen on vastattava hylkäyssignaalilla kaikkiin ulkoa vastaanottamiinsa sanomiin ja vapautettava omalta yksiköltä vastaanotetut signaalipuskurit.

Jaettaessa sanoma–ajuri kahteen osaan sen perusteella, välittääkö se yksikölle tulevia vai siltä lähteviä sanomia, saadaan kaksi yksinkertaista tilakonetta (kuvat 22 ja 23); vieraita pistoyksiköitä kuunteleva osa on karkeasti ottaen joko normaalitilassa välittäen tulevat sanomat oman yksikön prosesseille tai sitten se kieltäytyy reitityksestä.



Kuva 22 Sanoma–ajurin vieraita pistoyksiköitä palvelevan osan päätilat

Ulkoisilta pistoyksiköiltä saapuvat sanomat välitetään omalle solmutason palvelimelle heti käynnistyksen jälkeen sitä mukaa kun niitä saapuu, jotta käynnistys saadaan mahdollisimman nopeaksi.



Kuva 23 *Omalta yksiköltä ulospäin suuntautuvaa sanomaliikennettä palvelevan sanoma–ajurin osan päätilat*

Vieraille yksiköille sanomia välittävä osa odottaa aluksi yksittäisten yksiköiden rekisteröitymistä ja sen jälkeen lähettää niille tarkoitetut odottamassa olleet ja uudet sanomat. Vieraan yksikön poistuessa solmusta sanoma–ajuri jää odottamaan uudelleenrekisteröitymistä. Myös tämä jälkimmäinen osa reagoi irtikytkeytymiskäskyyn. Sanoma–ajurien välisessä liikennöinnissä on periaatteena, että ellei ulos lähetettyyn sanomaan saada vastapuolelta kuittausta, lähetys toistetaan yhden kerran. Kaikille yksiköille tarkoitettujen lähtevien sanomien jakelulistalta suodatetaan oma pistoyksikkö pois; tämä käytäntö sulkee pois mahdollisuuden helppoihin silmukointeihin testausta varten, mutta tekee kaikille yksiköille kohdistetut ryhmälähetykset solmutason palvelimille helpoiksi.

Topologilta saapuvia pistoyksikköilmoituksia ei ole suodatettu, joten räpsyvä eli hyvin nopeassa tahdissa vian vuoksi käynnistyvä ja sammuva pistoyksikkö aiheuttaa aikavalvotun sanoman osalta ylimääräistä kuormitusta lähettävien pistoyksiköiden puolella. Ongelma ei kuitenkaan ole kovin vakava, sillä todennäköisyys että räpsy-

välle yksikölle ollaan lähettämässä aikavalvottua sanomaa on suhteellisen pieni ottaen huomioon, että aikavalvontaa käytetään lähinnä solmun käynnistyessä. Mahdollisen suodatusajan tulisi olla lyhyt, jottei syntyisi tarpeetonta viivettä siinä tilanteessa, kun solmuun lisätään pistoyksikkö joka ennen takalevyn liittimeen kunnolla kytkeytymistään ehtii räpsähtää päälle ja pois, eikä kovin lyhyestä suodatuksesta ole vastaavaa hyötyä.

Ylläpitäessään tietoa läsnä olevista yksiköistä ajuri kirjaa ainoastaan topologin antamat ilmoitukset; hylkäysviestien ja epäonnistuneiden lähetysten ottaminen syötteinä mukaan voisi mahdollisesti johtaa sekaannuksiin jotka voivat syntyä eri kautta kulkevien tietojen nopeuseroista.

Reaktiot ulkoisiin herätteisiin:

Pistoyksikkötapahtuma

Heräte päivittää läsnä olevista yksiköistä pidettävää listaa. Koska ei ole mahdollista, että kerran solmutasolta pois kytkeytynyt yksikkö myöhemmin palaisi mukaan toimintaan, heräte voidaan jättää huomiotta sen jälkeen kun on saatu *Irtikytkeytymiskäsky*.
Yksikön saapuessa paikalle tarkistetaan onko sitä odottamassa yhtään sanomaa ja suoritetaan mahdollinen lähetys.

Irtikytkeytymiskäsky.

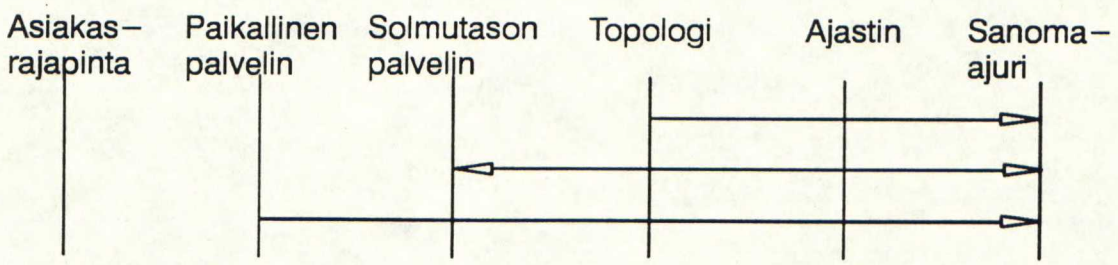
Heräte muuttaa prosessin tilan; lähetyslistat pyyhitään tyhjiksi, muilta yksiköiltä tuleviin sanomiin vastataan lähettämällä *Pistoyksiköiden välinen sanomanhylkäys* ja olamalta yksiköltä tulevat signaalit vapautetaan.

Pistoyksiköiden välinen sanoma

Mikäli oma yksikkö on mukana solmutason toiminnassa, sanoma kuitataan hyväksytyksi ja välitetään omalle solmutason palvelimelle, muutoin vastataan hylkäysilmoituksella.

Sanoma solmutason palvelimelta

Sanoma lähetetään kaikille läsnä oleville vastaanottajiksi määritellyille yksiköille odottaen kuittaukset. Mikäli sanomalle on määritelty aikavalvonta, se lisätään odottavien sanomien listaan.



Kuva 24 Sanoma–ajurin signalointisidokset oman pistoyksikön sisällä

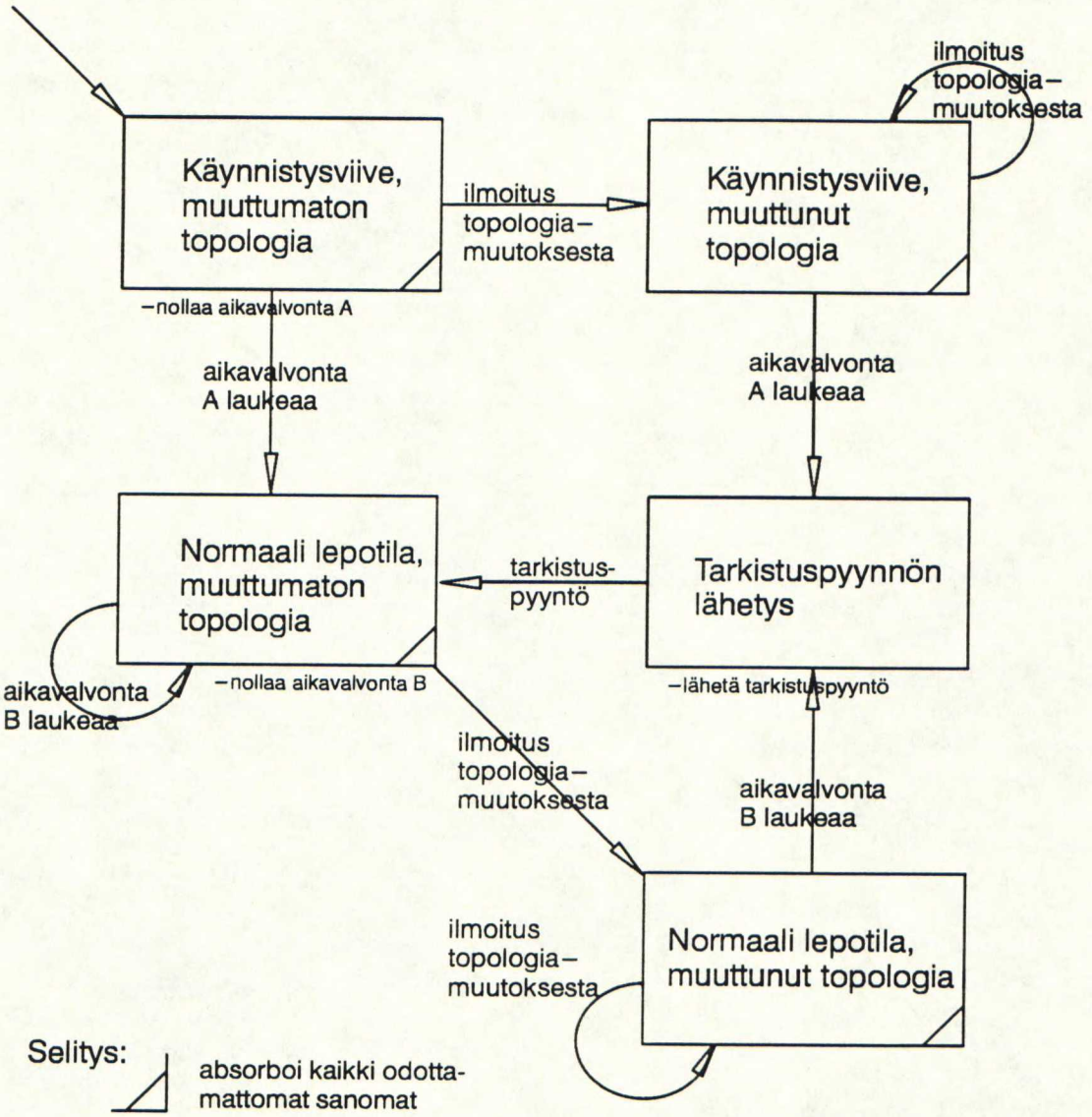
Kuva 24 esittää pistoyksikön sanoma–ajurin signalointisidokset oman yksikön muihin tiedonvarmistusjärjestelmän prosesseihin; lisäksi ajuri tarpeen vaatiessa kommunikoi kaikkien muiden solmussa läsnä olevien pistoyksiköiden sanoma–ajurien kanssa.

5.3.7 Täydennyslevitysajastin

Täydennyslevitysajastimelle saapuva *Ilmoitus muuttuneesta varmistustopologiasta* saattaa käynnistää koko solmun tiedostojen läpikäymisen. Laajassa solmussa toimenpiteen aiheuttama kuorma vie resursseja suhteellisen paljon erityisesti jos monta yksikköä selvittelee tiedostojensa levitystilannetta samanaikaisesti, joten tarkastukset lomitetaan ajan suhteen siten, että kutakin solmun pistoyksikköä varten on varattu aikaväli, jolloin se voi suorittaa tiedostotarkastelun. Oletuksena on, että solmun pistoyksiköt käynnistyvät likimain yhtäaikaisesti eli että ne ovat olleet solmussa läsnä sen käynnistyshetkellä. Kukin ajastin pitää käynnistyksen yhteydessä yksikön solmun sisäisestä sijainnista riippuvan tauon, joka määrää eri yksiköiden levitystarkastelun käynnistysjärjestyksen; tauon pituus on porrastettu siten, että kukin yksikkö on todennäköisesti ehtinyt suorittaa tarkastustoimensa loppuun saakka ennen seuraavan yksikön aktivoitumista. Ensimmäisen tarkastushetken alun koitettua kaikki pistoyksiköt pitävät keskenään samanmittaisen tauon ennen uutta tarkastusta, joten koko solmun tarkastussyklin pituus määräytyy siihen mahdollisesti kuuluvien yksiköiden lukumäärän mukaan.

Ratkaisun geneerisyyden varmistamiseksi aikaporrastusta määriteltäessä käytetään suurinta yksiköiden lukumäärää joka on mahdollista kalustaa solmuun, eikä läsnä olevien yksiköiden määrää. Vajaaksi kalustetussa solmussa tämä tarkoittaa, että tiedostojen levityksen aukottomuutta ei tarkasteta niin lyhyin aikavälein kuin mahdollista, mutta toimenpide ei myöskään pääse kuormittamaan solmua kriittisesti. Täysin kalustetun solmun tapauksessa kuormitus on jatkuvasti vajaata solmua korkeampi johtuen siitä että tarkistavia ja tarkistettavia yksiköitä on paljon ja paljon täydennystarvetta paljastavat tarkastukset voivat liu'uttaa yksiköiden aktiiviset vaiheet päällekkäin, joten aikaportaat on syytä määrätä suurimman yksikköluvun mukaan. Käsiteltävänä olevan järjestelmän kohdalla laskentaperusteina on käytetty 20 pistoyksikköä ja kahden minuutin aikaväliä, joten kunkin asennuskehikon korttipaikassa sijaitsevan pistoyksikön kohdalle tarkistusvuoro sattuu 40 minuutin välein. Aikaväli kannattaa pitää kohtuullisen suurena, etteivät syystä tai toisesta aiheutuvat

toistuvat solmukonfiguraation muutokset johda tiedonvarmistusjärjestelmän aiheuttamaan suureen kuormitukseen. On syytä myös muistaa, että tietyn yksikön kohdalle sattunut tarkastusaikaväli käytetään vain, jos yksikön näkemä ympäristö on muuttunut edellisen tarkistuksen jälkeen. Heti pistoyksikön käynnistymisen jälkeen tapahtuvat turhat tarkastukset on eliminoitu siten, että ensimmäisestä topologiapäätöksestä ei saada muutosilmoitusta.



Kuva 25 Ajastimen sisäiset tilat

Käytetty ratkaisu on toimiva edellyttäen ettei millään yksiköllä ole odotettua suurempia täydennyslevitystarpeita ja ettei solmussa ajan myötä käynnistetä kovin

montaa yksikköä sellaisina hetkinä, että niiden tarkistusaikavälit sattuvat päällekkäin jo käytössä olevien aikavälien kanssa; kukin ajastin on toisista riippumaton ja alkaa laskea aikaa siitä hetkestä, kun oma pistoyksikkö kytketään käyttöjännitteeseen, koska ajastus perustuu käyttöjärjestelmältä saatavien kellopulssien laskemiseen.

Reaktiot ulkoisiin herätteisiin:

Prosessi odottaa vain *Ilmoitus muuttuneesta varmistustopologiasta* –signaaleja, kaikki muut herätteet hylätään. Ajastimen lauettua prosessi lähettää solmutason palvelimelle *Tarkistuspyyntö* –signaalin edellyttäen, että edellisen lähetyksen jälkeen on saatu ainakin yksi topologiamuutosilmoitus.

5.4 Järjestelmän toiminta oliotasolla

Ennen olioiden välisen toiminnan kuvaamista määritellään järjestelmän tärkeimmät oliot, jotka voidaan jakaa prosessien toimintaa ylläpitäviin sekä toimintaa tukeviin yleisiin ja tietokantaolioihin. Lisäksi ulkoisia ohjelmistokomponentteja käsitellään olioina.

5.4.1 Oliomäärittely

Prosessien toimintaa ylläpitävät ja ohjaavat oliot:

Asiakasrajapinta: Vaatimusmäärittelystä seuraava luokka.

Paikallinen palvelin: Palvelinprosessin ulkoisena rajapintana toimiva olio, joka vastaa prosessille asetettujen vaatimusten toteuttamisesta.

Solmutason palvelin: Solmutason palvelimen rajapintaolio.

Sanoma–ajuri: Pistoyksiköiden välisen kommunikoinnin matalan tason toteutuksesta vastaava olio.

Topologi: Topologin rajapintaolio.

Täydennyslevitysajastin: Ajastinprosessin rajapintaolio.

Tukioliot:

Sijaintipalvelu: Olio, joka tuntee oman yksikön sijainnin ja tyypin solmussa.

Nimipalvelu: Tiedostojen nimien muodostamiseen ja tulkitsemiseen erikoistunut olio.

Symbolipalvelu: Tämä olio osaa muodostaa ja tulkita solmun sisällä ainutkertaisia IUCM–osoitesymboleita.

Sanomaolio: Olio, jota käytetään apuna muodostettaessa lähetettäessä ja tulkittaessa prosessien välisiä käyttöjärjestelmäsignaaleja.

Tietokantaoliot:

Tietokantaoliot ovat muunnelmia perusoliosta, jota voidaan käyttää erilaisten tietojen tilapäiseen säilyttämiseen ja hakuihin. Koska tietokannat sisältävät samoja perusominaisuuksia, yhteinen osa on toteutettu metaluokkana, josta on peritty eri-

koisominaisuuksien toteutuksen sisältävät erikoistuneita olioita vastaavat aliluokat.

Identiteettitietokanta: Solmun tai pistoyksikön identiteetin päätösvaiheessa tarvittava toiminnallinen tietovarasto.

Ajastintietokanta: Ajastinpalveluiden perustana toimiva olio, jolta voi pyytää ajastukseen liittyviä palveluja.

Pistoyksikkötietokanta: Olio, joka ylläpitää sille annettavien herätteiden perusteella tietoa pistoyksiköiden ominaisuuksista.

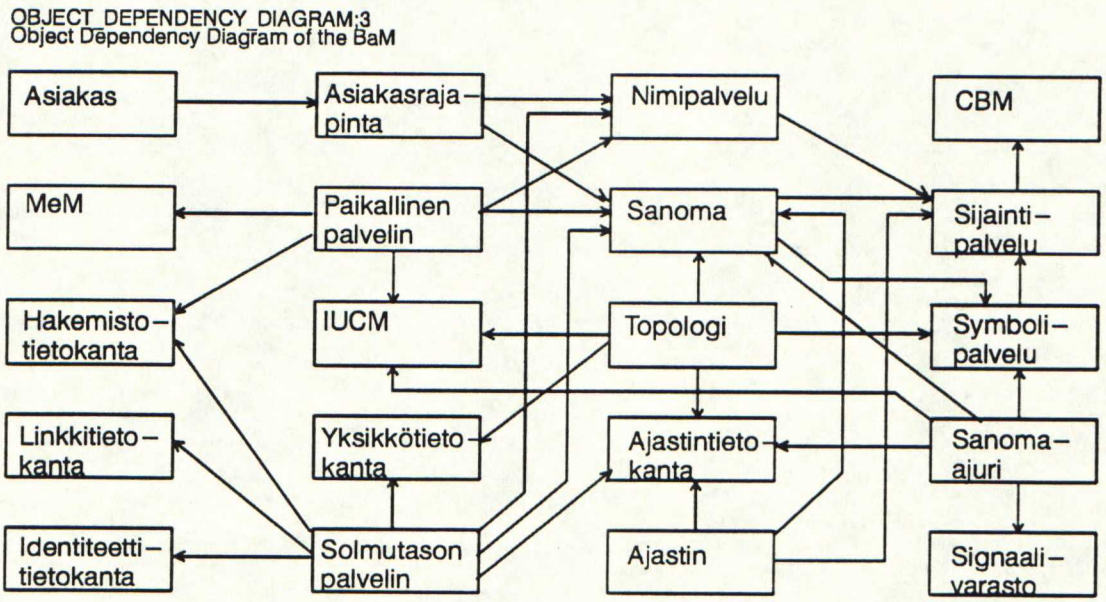
Signaalivarasto: Olio, joka säilöö viittauksia käyttöjärjestelmäsignaaleihin, joiden elinkaari jatkuu vielä ensimmäisen prosessointikerran jälkeen.

Tiedostojen levityslinkkitietokanta: Levityslinkkiyhdistelmien säilömiseen ja muokkaamiseen erikoistunut olio.

Hakemistotietokanta: Tiedostohakemistopalveluihin erikoistunut olio.

Tukikomponentit:

Varmistusjärjestelmän tuekseen tarvitsemat ohjelmistokomponentit ja niiden vastuualueet on määritelty mm. kuvassa 13 sivulla 34.



Kuva 26 Olioriippuvuuskaavio

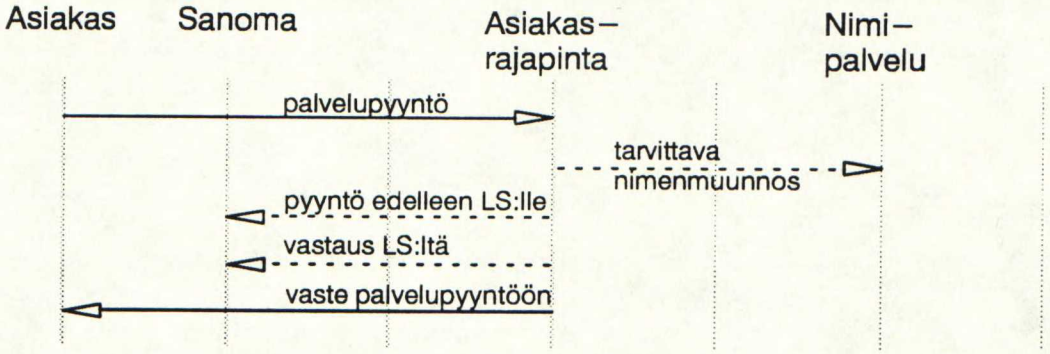
Suurin osa järjestelmän olioista tarvitsee tehtävänsä suorittamiseksi apua muilta olioilta; nämä riippuvuudet muodostavat kuvan 26 mukaisen verkon, jonka kautta oliot kytkeytyvät toisiinsa.

5.4.2 Olioiden toiminta

Järjestelmän toiminta oliotasolla käydään läpi prosessi kerrallaan aloittaen asiakasrajapinnasta.

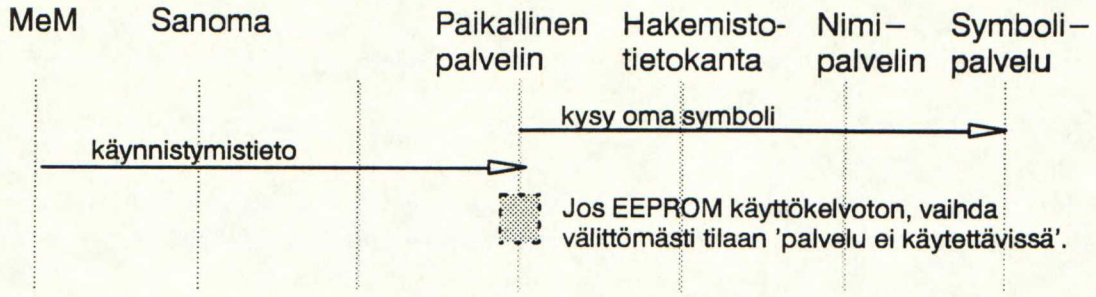
Seuraavissa oliotoimintakaavioissa pystyviivat kuvaavat olioita ja viivojen väliset nuolet metodikutsuja. Loogisia kokonaisuuksia kokoamaan käytetään suorakulmioita. Katkoviiva kuvaa ehdollisuutta eli esimerkiksi että tietty heräte lähetetään vain jos jokin ehto toteutuu. Pystyakseli kuvaa suoritusjärjestystä eli suhteellista aikaa.

Paikallisesta palvelimesta käytetään tarvittaessa lyhennettä LS (*Local Server*) ja solmutason palvelimesta lyhennettä IUS (*Inter–unit Server*).

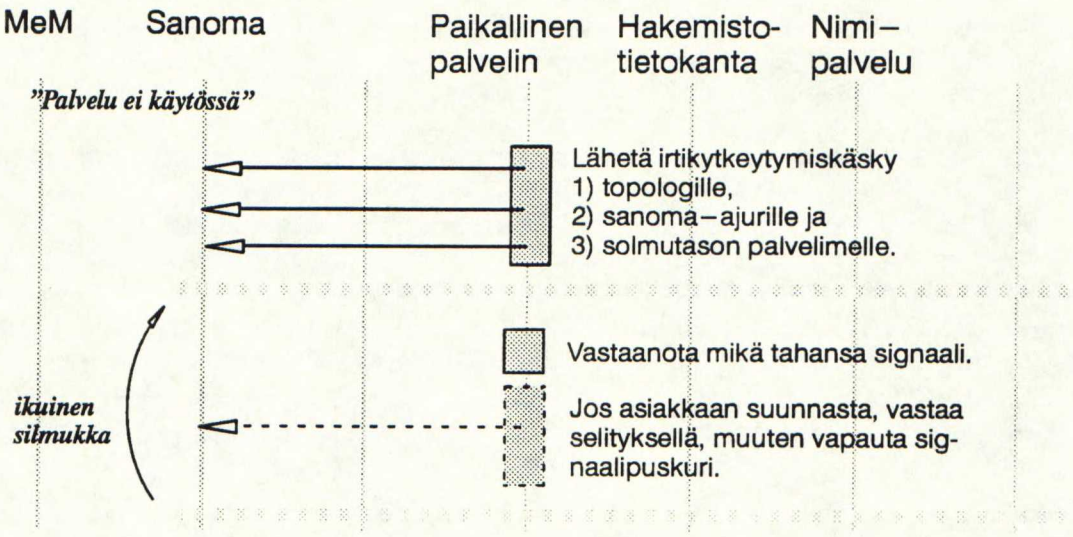


Asiakasrajapinta siis yksinkertaisesti vain odottaa asiakkaalta tulevia palvelupyyntöjä ja ohjaa ne yksi kerrallaan sanomaolion avulla paikalliselle palvelimelle, jonka toiminta solmun käynnistymishetken jälkeen koostuu kolmesta eri vaiheesta: käynnistymisestä, levitettävien tiedostojen päivityksestä ja normaalista toiminnasta.

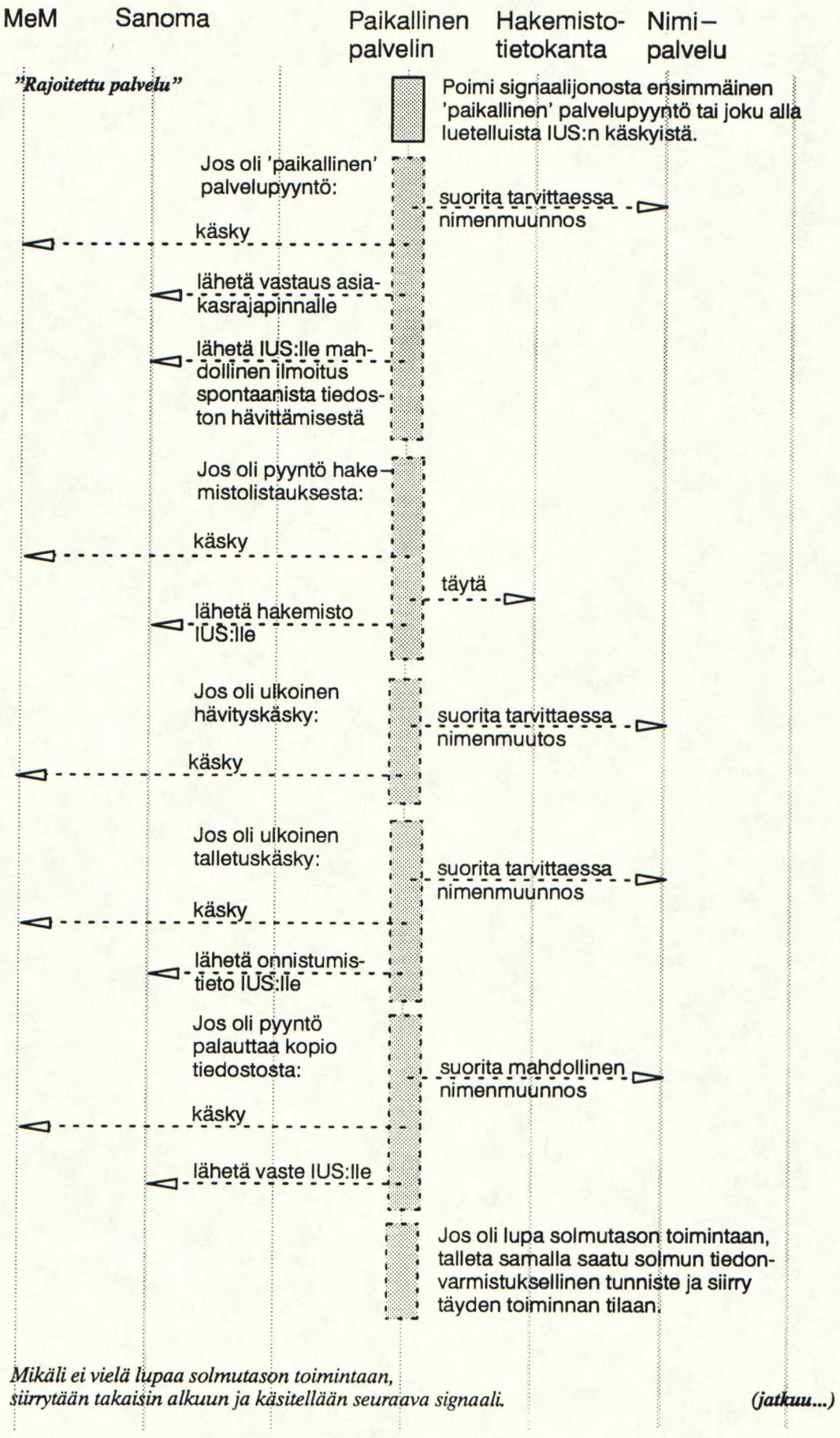
Aluksi on odotettava EEPROM:n käyttöä valvovan MeM:n käynnistymistä eikä asiakkaita voida palvella.



Jos EEPROM on käyttökelvoton, palvelin joutuu lopullisesti tilaan, jossa se ei voi tarjota palveluita:



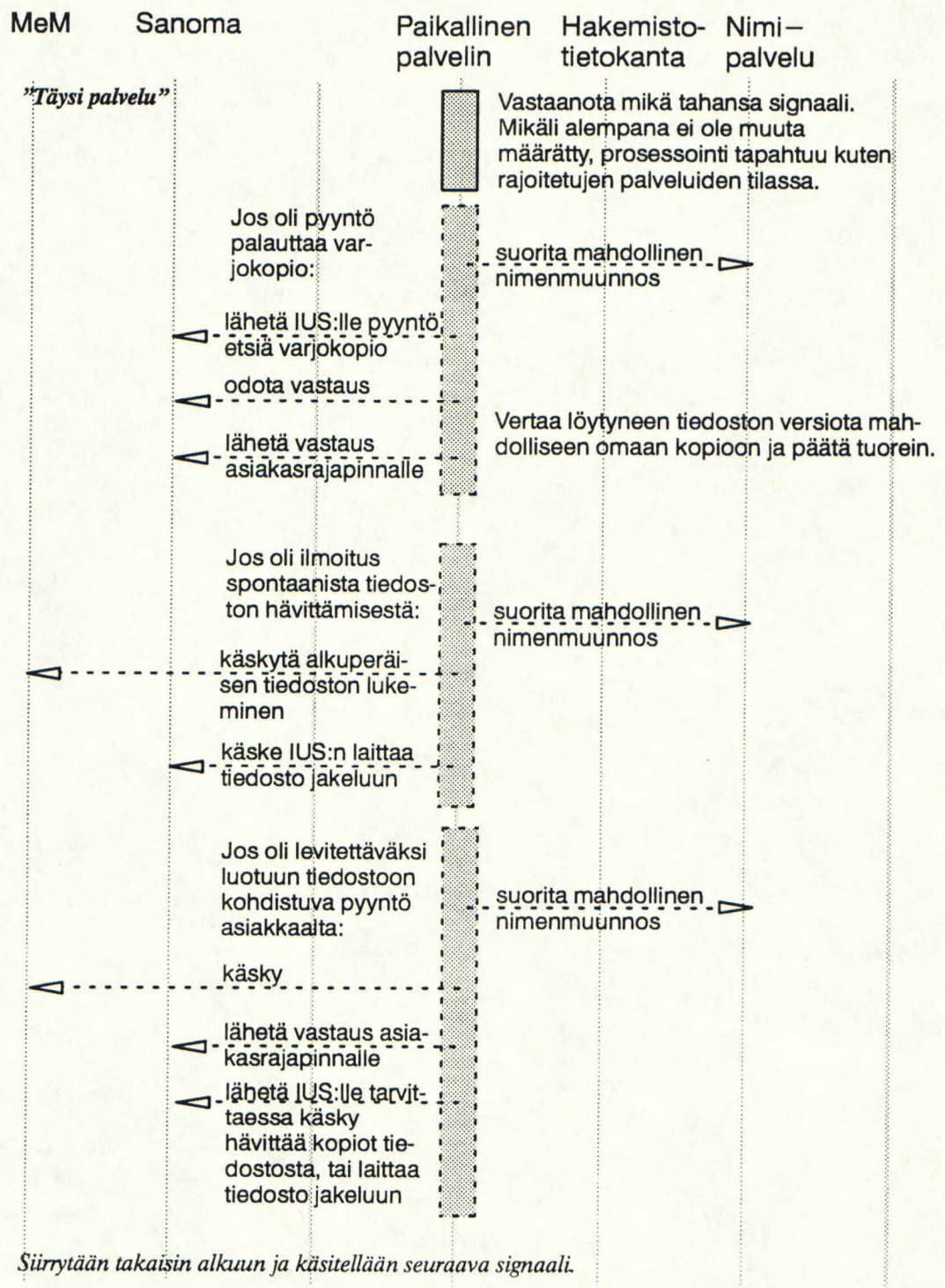
Mikäli MeM käynnistyi normaalisti, paikallinen palvelin siirtyy tilaan, jossa se täyttää asiakkaiden esittämiä paikallisiksi luotuihin tiedostoihin kohdostuvia pyyntöjä samanaikaisesti kun solmutason palvelin päivittää levitettäviksi luotuja tiedostoja käskyttäen paikallista palvelinta.



Mikäli ei vielä lupaa solmutason toimintaan, siirrytään takaisin alkuun ja käsitellään seuraava signaali.

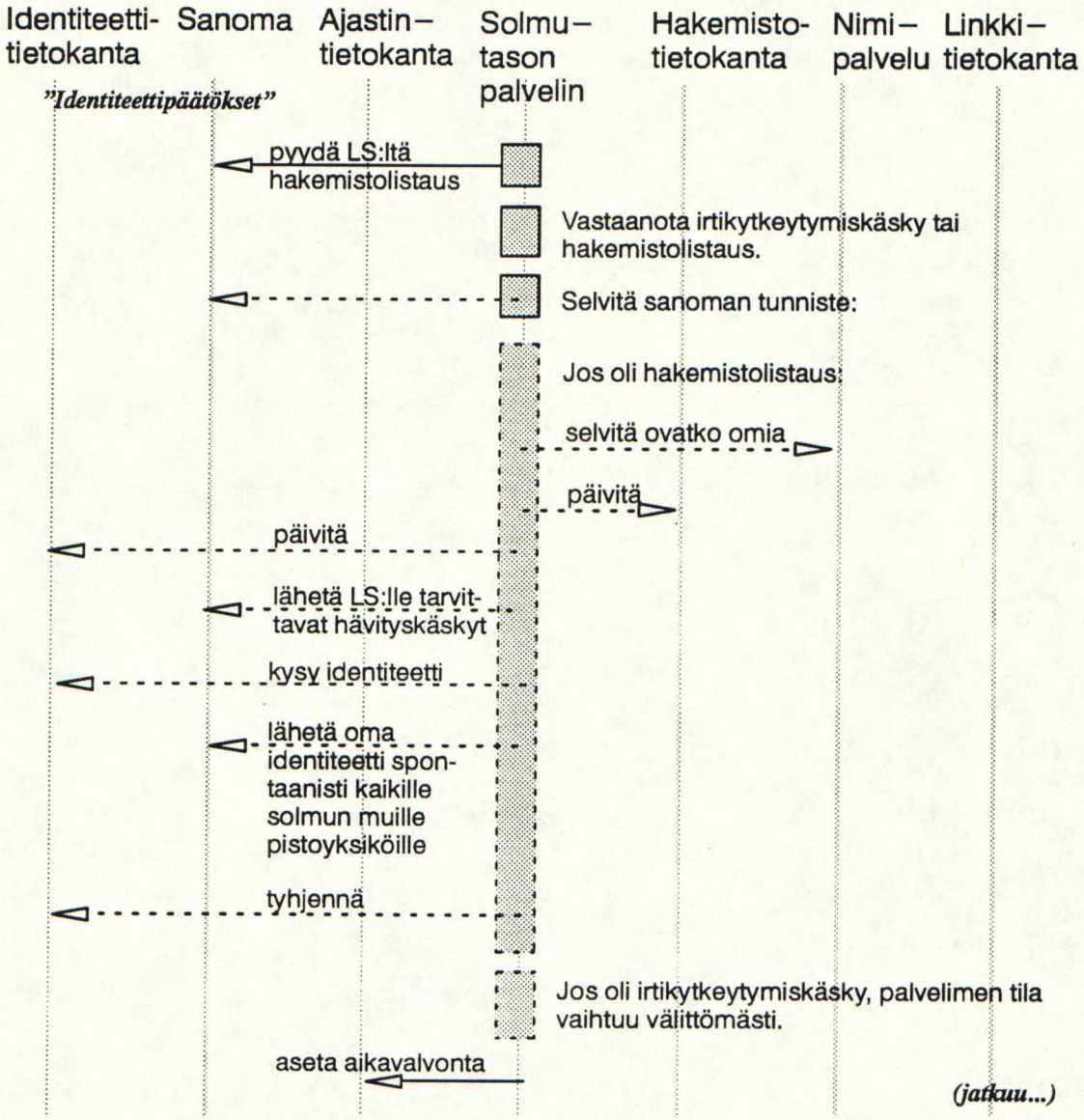
(jatkuu...)

Heti kun solmutason palvelin on luvan lähettämällä ilmaissut, että pistoyksikön kaikki tiedostot ovat ajantasalla, paikallinen palvelin siirtyy varsinaiseen päätilaansa, jossa se käsittelee kaikki asiakkailta ja solmutason palvelimelta tulevat pyynnöt:



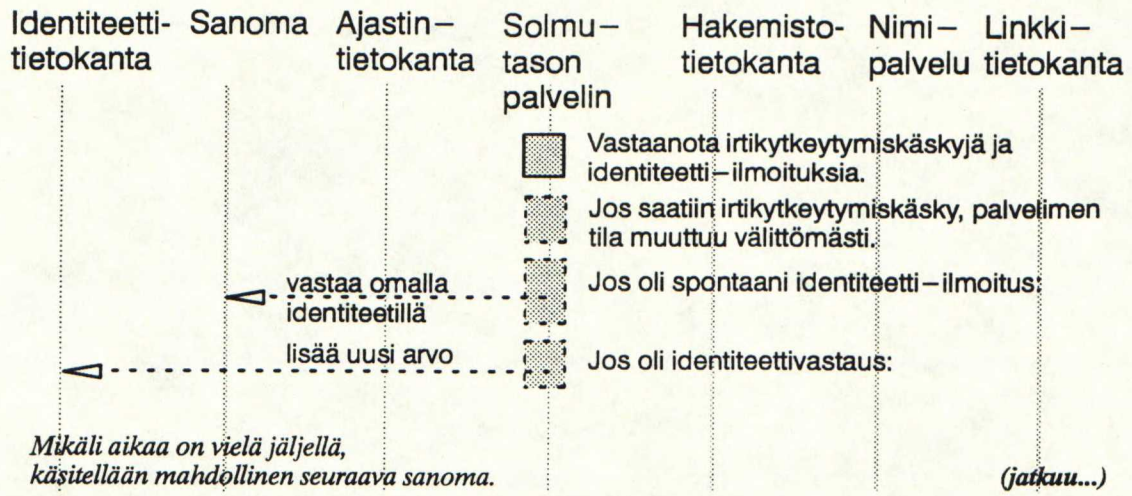
Paikallinen palvelin pyytää solmutason palvelimelta apua kaikkiin tehtäviin, joita se ei itse pysty suorittamaan. Myös solmutason palvelimen elinkaari koostuu peräkkäisistä vaiheista:

Aluksi palvelimen on päätettävä oman pistoyksikön tiedonvarmistuksellinen identiteetti ja osallistuttava sen avulla solmutason identiteetin päättämiseen.



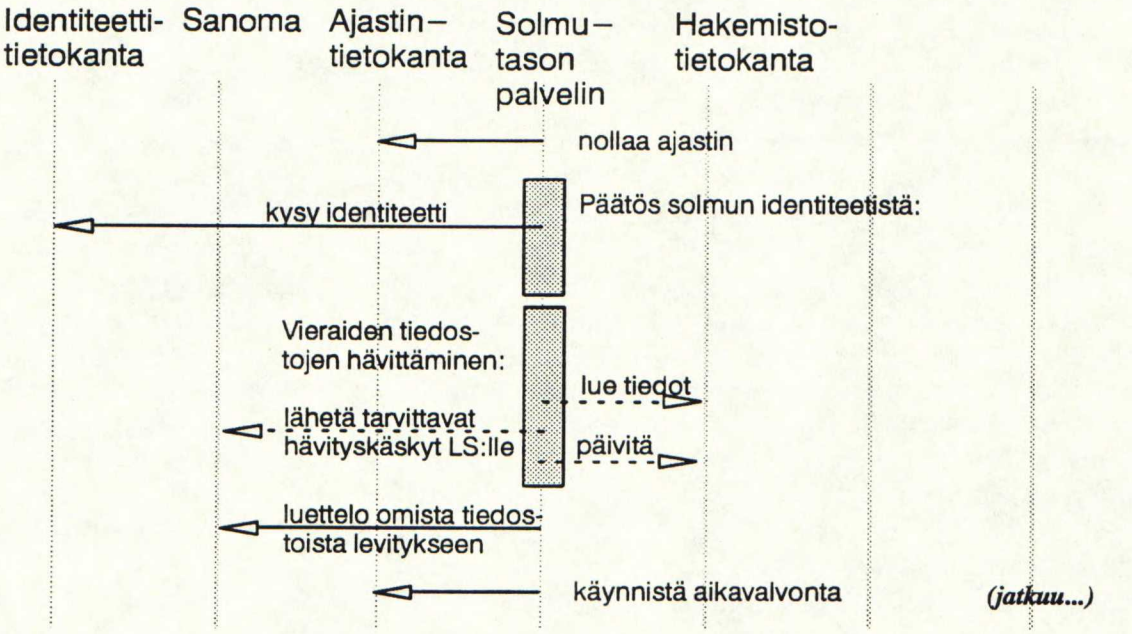
Jos paikallinen palvelin oli EEPROM:n käyttökelvottomuuden takia lähettänyt irtikytkeytymiskäskyn, solmutason palvelin siirtyy lopullisesti lepotilaan: palvelin tyhjentää koko ajan signaalijonoaan, muttei prosessoi käskyjä vaan vapauttaa signaalipuskurit.

Mikäli toiminta sai jatkua normaalina, palvelin on saanut päätettyä oman yksikkönsä identiteetin ja käynnistänyt aikavalvonnan kuulostellakseen muiden yksiköiden identiteetti–ilmoituksia määräajan.



Aikavalvonnalla taataan kaikille yksiköille minimiaika, jonka kuluessa niiden tulee vastata, jotta vastaus ehditään hyväksyä tunnistepäätöksen pohjaksi. Moniajoympäristössä ei voida taata minkään prosessin pääsevän ajoon heti kun vieraalta yksiköltä saapuu vastetta edellyttävä sanoma, joten aikavalvonta on säädettävä tarpeeksi pitkäksi tavanomaista solmun käynnistyskuormitusta varten.

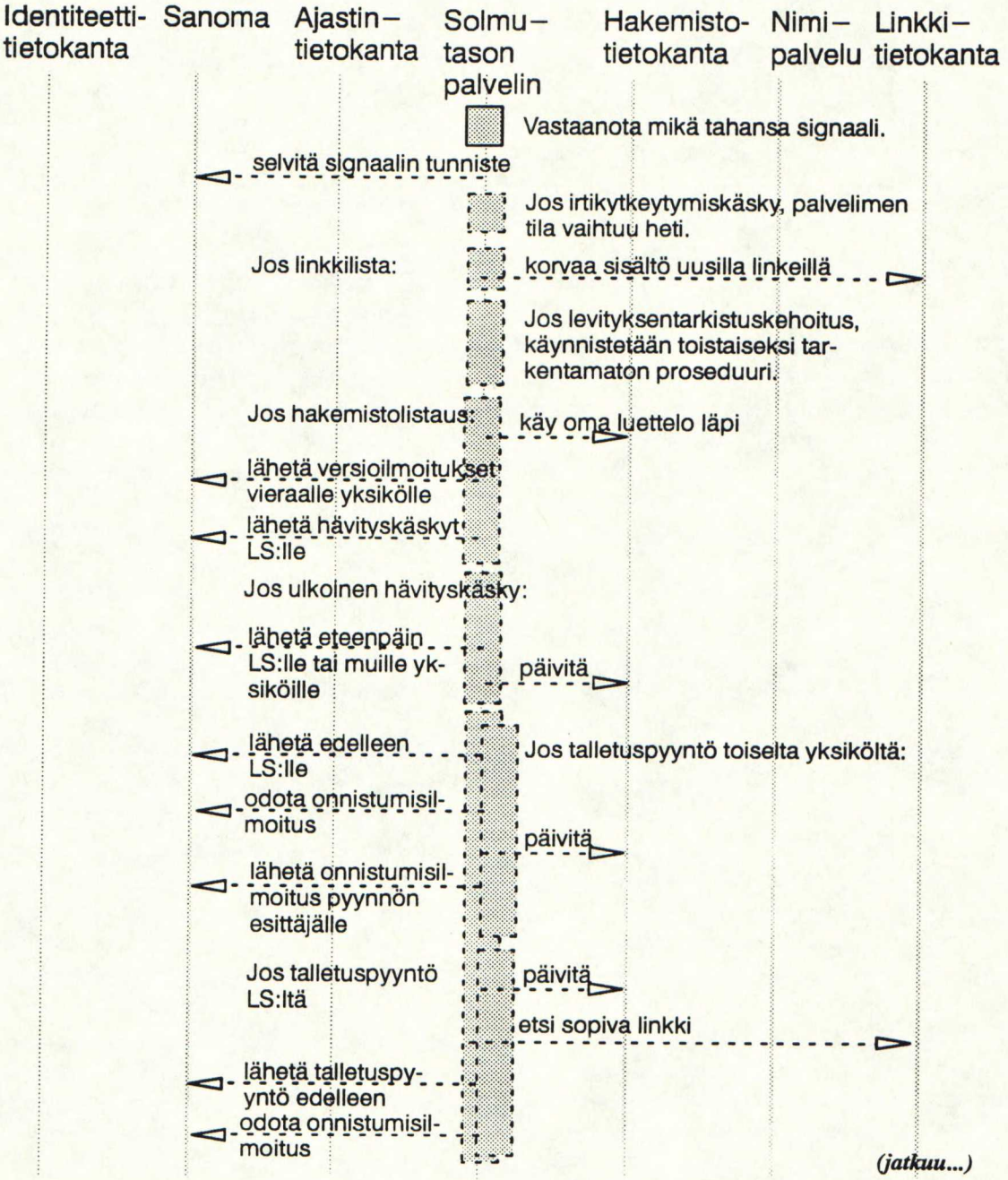
Aikavalvonnan lauettua palvelin tekee ajoissa saapuneiden identiteetti=ilmoitusten perusteella oman päätöksensä solmun identiteetistä ja käskyttää paikallisen palvelimen hävittämään kaikki ympäristöön kuulumattomat tiedostot:

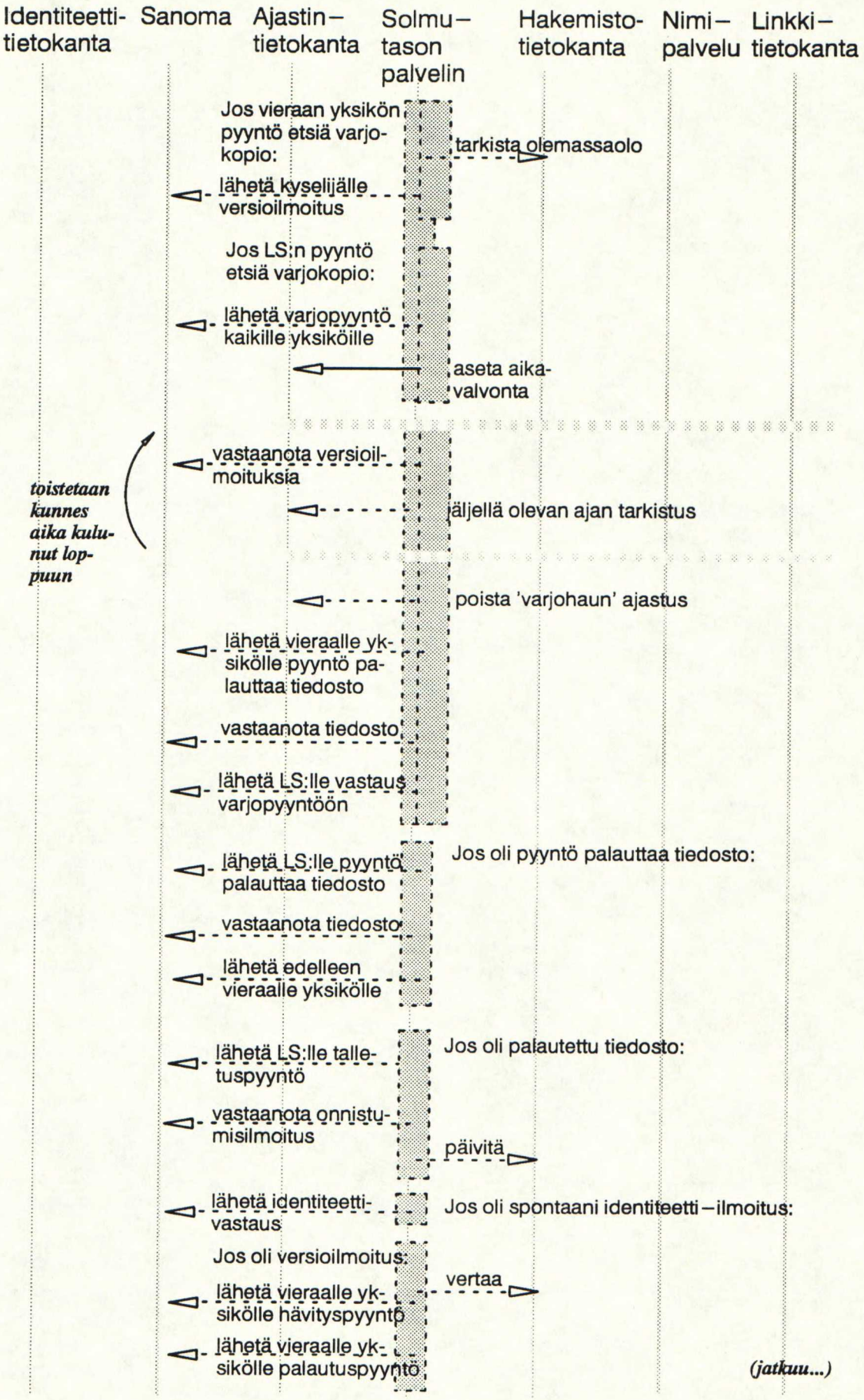


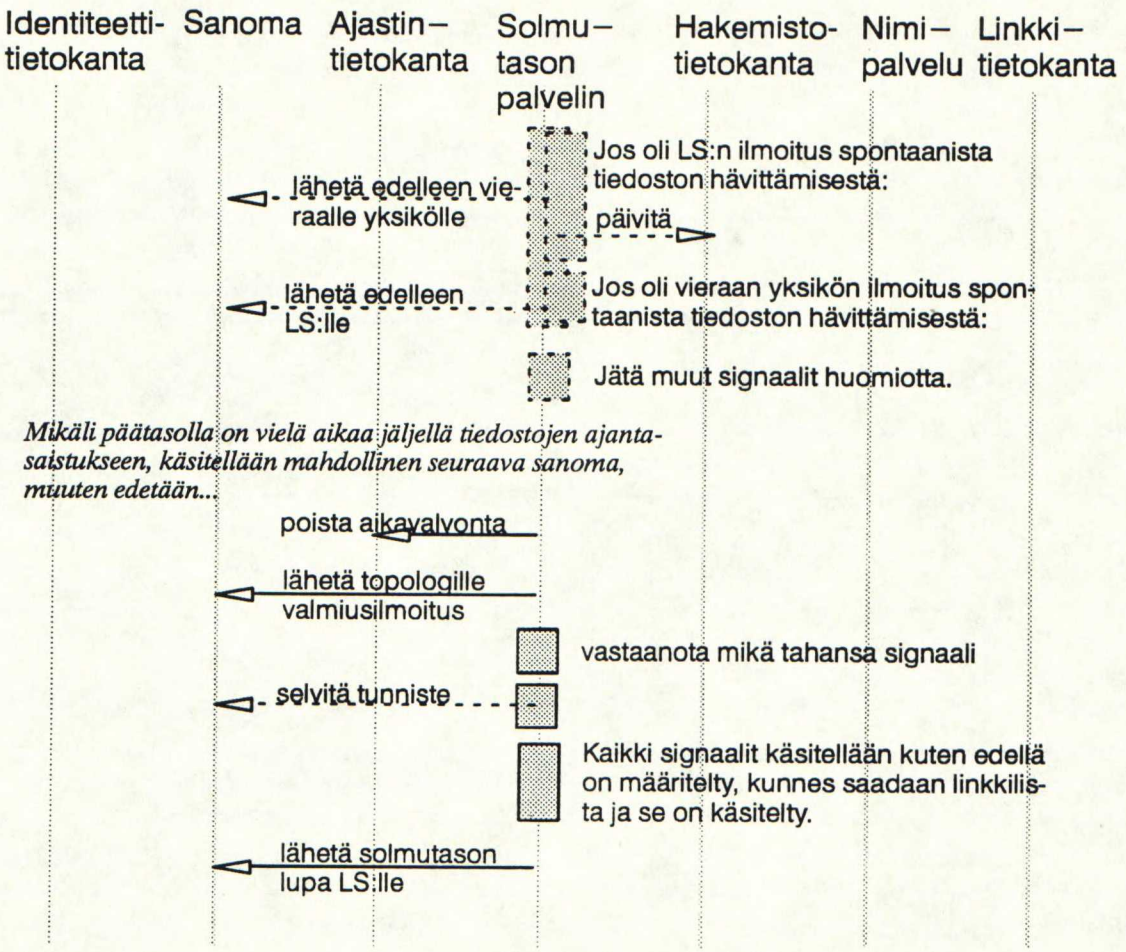
Solmutason palvelin on järjestelmän prosesseista kaikkein mutkikkain, mikä ilmenee mm. rinnakkaisten toiminteiden suorittamisena sellaisten tilanteiden välttämiseksi, joissa kahden tai useamman eri yksikön solmutason palvelimien toisilleen lä-

hettämät vastetta vaativat sanomat menisivät matkalla ristiin ja jäisivät vastaanottajan signaalijonoon käsittelemättä, koska vastausta edelliseen sanomaan ei vielä ole saatu.

Identiteettipäätöksen synnyttyä solmutason palvelin päivittää omalta yksikön levitettäviksi luodut tiedostot ajantasalle auttaen samalla muiden yksiköiden solmutason palvelimia vastaavissa toimissa. Jotta tiedostojen päivityksen epäonnistuminen vikatilanteessa ei pääse lukitsemaan koko tiedonvarmistuspalvelua, rajoitetaan päivittämiseen varattu aika aikavalvonnalla, joka asetettiin solmuidentiteettipäätöksen tekemisen jälkeen.

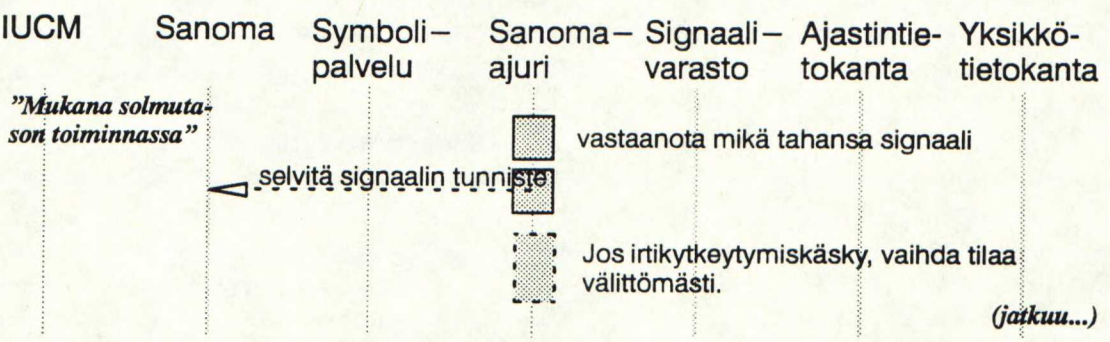


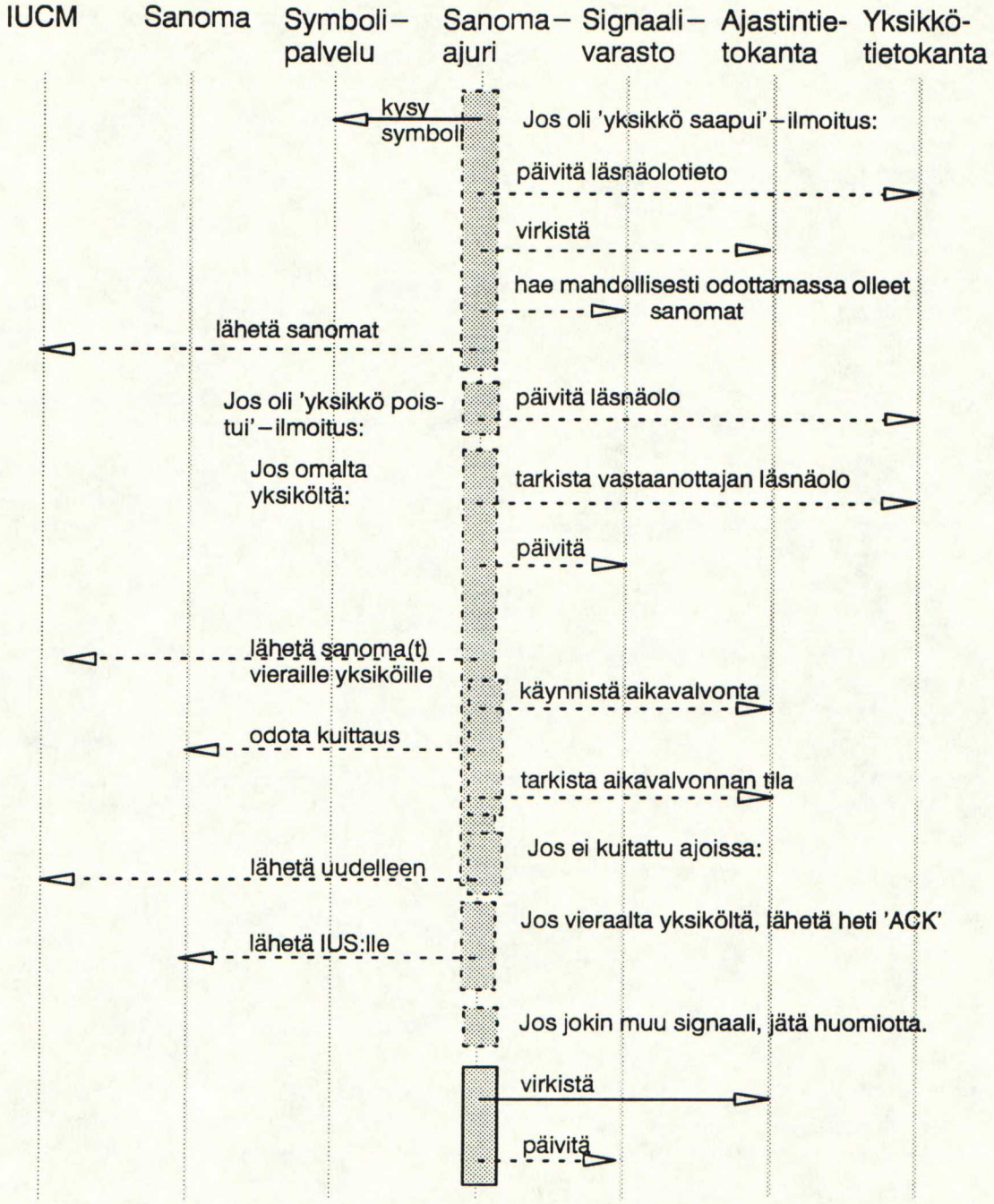

Identiteetti-tietokanta
Sanoma
Ajastin-tietokanta
Solmu-tason palvelin
Hakemisto-tietokanta
Nimi-palvelu
Linkki-tietokanta
Jos vieraan yksikön pyyntö etsiä varjo-kopio:
tarkista olemassaolo
lähetä kyselijälle versioilmoitus
Jos LS:n pyyntö etsiä varjokopio:
lähetä varjopyyntö kaikille yksiköille
aseta aika-valvonta
vastaanota versioil-moituksia
toistetaan kunnes aika kulu-nut lop-puun
jäljellä olevan ajan tarkistus
poista 'varjohaun' ajastus
lähetä vieraalle yk-sikölle pyyntö pa-lauttaa tiedosto
vastaanota tiedosto
lähetä LS:lle vastaus varjopyyntöön
Jos oli pyyntö palauttaa tiedosto:
lähetä LS:lle pyyntö palauttaa tiedosto
vastaanota tiedosto
lähetä edelleen vieraalle yksikölle
Jos oli palautettu tiedosto:
lähetä LS:lle talle-tuspyyntö
vastaanota onnistu-misilmoitus
päivitä
Jos oli spontaani identiteetti-ilmoitus:
lähetä identiteetti-vastaus
Jos oli versioilmoitus:
vertaa
lähetä vieraalle yk-sikölle hävityspyyntö
lähetä vieraalle yk-sikölle palautuspyyntö
(jatkuu...)



Nyt oman pistoyksikön tiedostot on ajantasaistettu ja palvelin siirtyy varsinaiseen päätilaan, jossa huolehditaan uusien tiedostoversioiden varmistamisesta ja täydennetään muuttuvasta ympäristöstä johtuvia vajavaisuuksia tiedostojen levityksessä; kaikki herätesignaalit käsitellään kuten edellä on määritelty.

Seuraavaksi käsitellään sanoma–ajurin toiminta; yksikön käynnistyessä ajuri kysyy oman IUCM–symbolinsa symbolipalvelulta ja alustaa IUCM:n muilta pistoyksiköiltä tulevia sanomia varten, minkä jälkeen ajuri toimii seuraavasti:



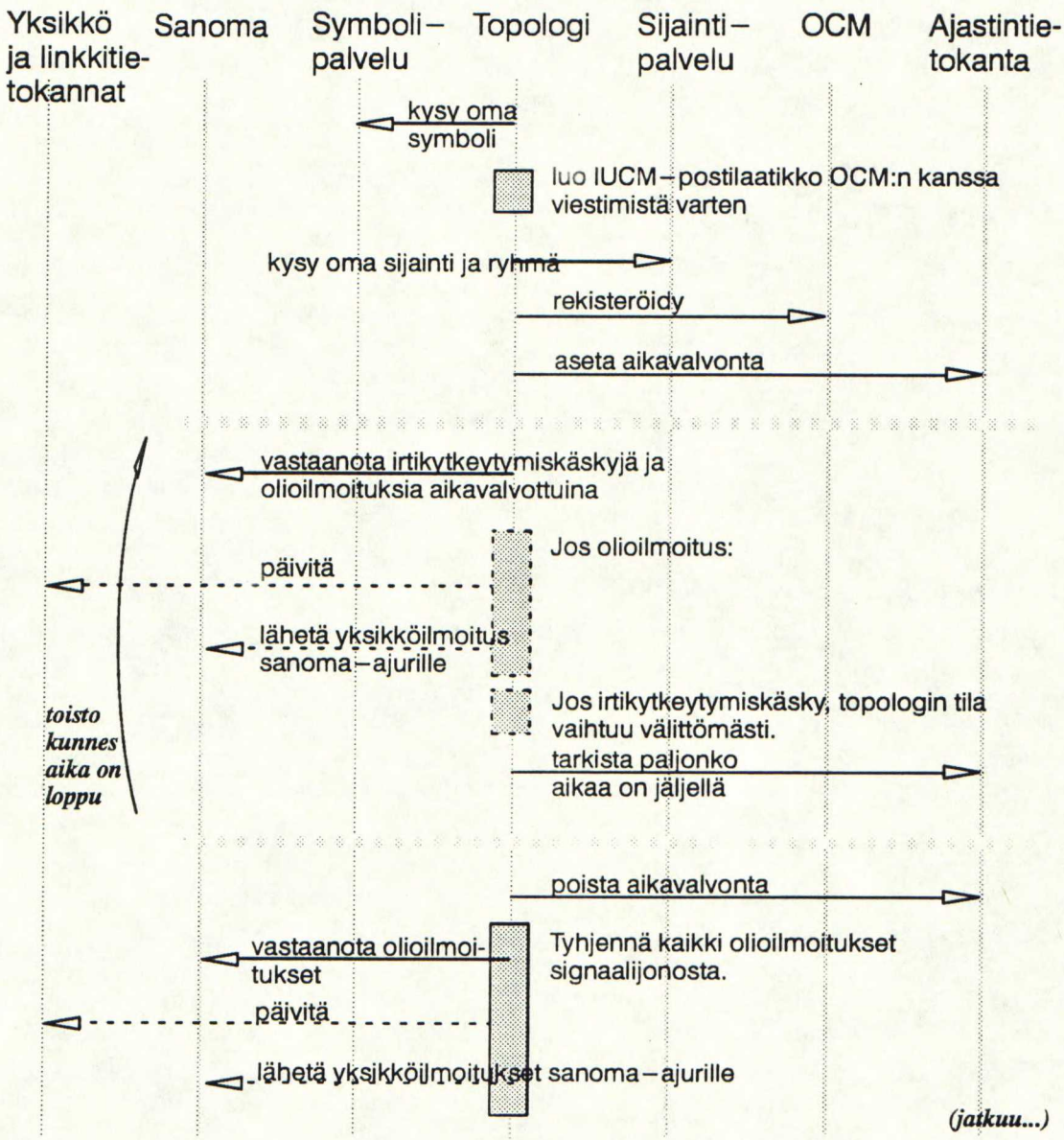


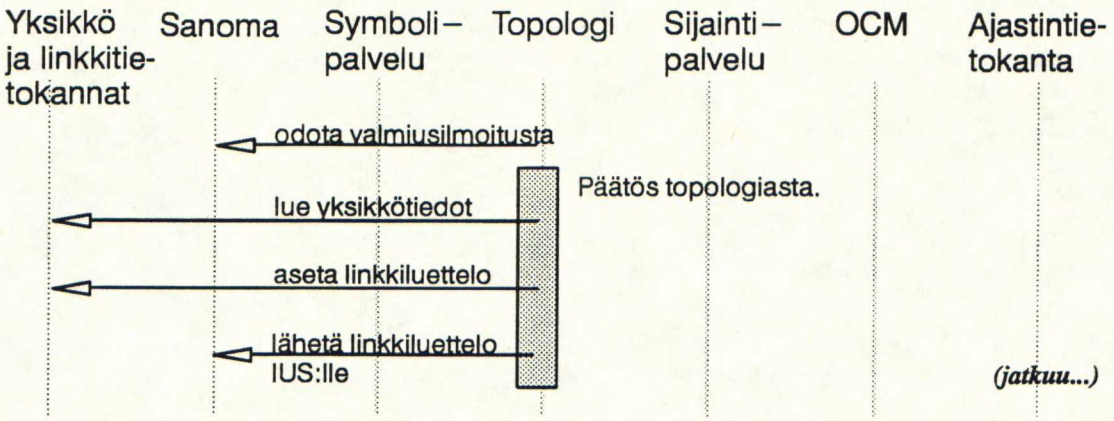
Irtikytkeytymiskäskyn saatuaan ajuri lakkaa välittämästä sanomia kumpaankaan suuntaan, tyhjentää muistinsa ja ryhtyy tyhjentämään signaalijonoaan.

Tiedostojen levittäminen solmun eri pistoyksiköille on suorituskykyongelmien välttämiseksi suoritettava harkiten ja ympäristömuutoksiin reagoiden, joten solmutason palvelin vaatii tuekseen vielä topologin, joka hoitaa ympäristön seurannan, levitysstrategian suunnittelun ja sen käskyttämisen palvelimelle.

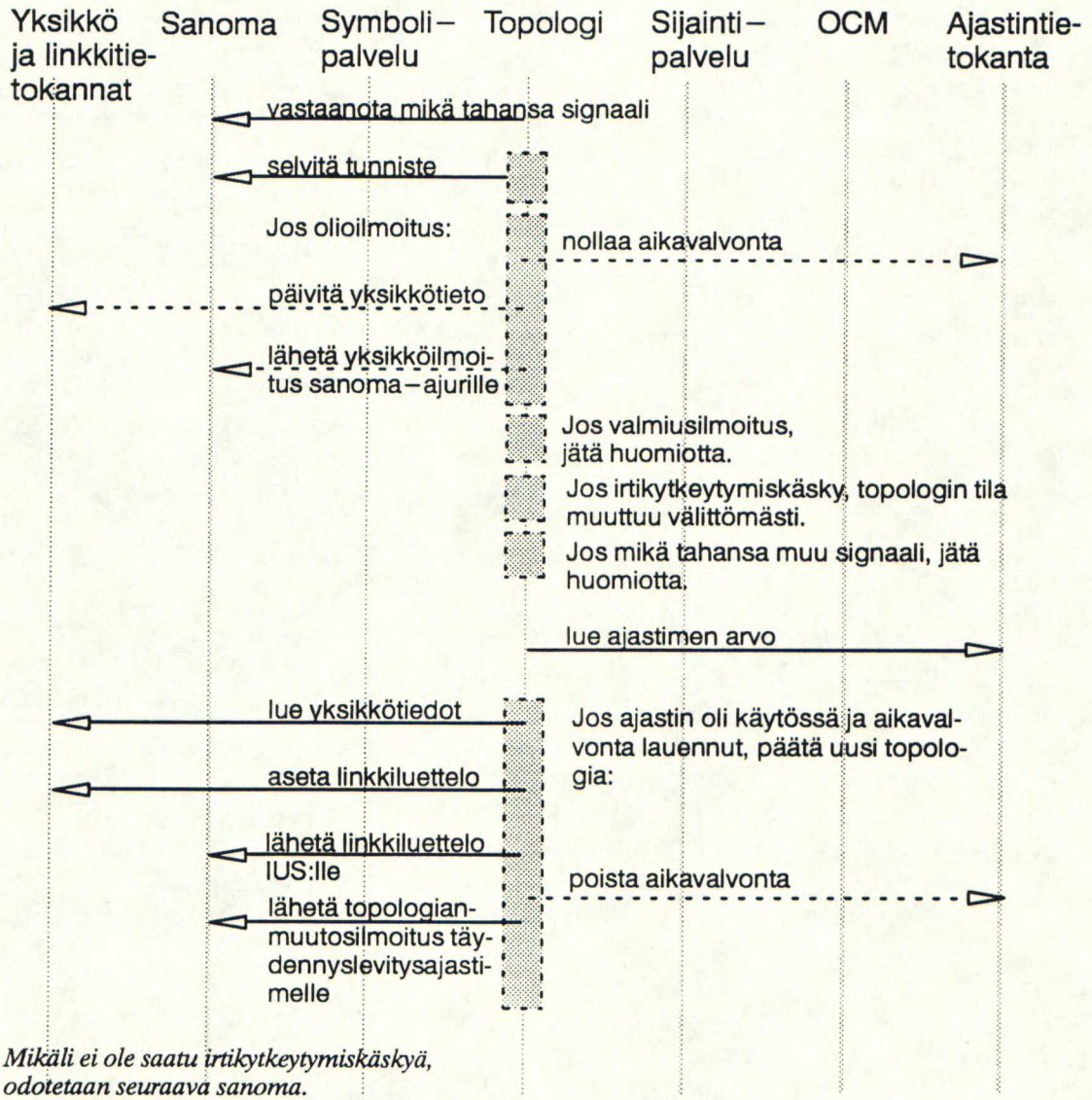
Topologi tukeutuu voimakkaasti OCM:iin, jolle se käynnistyessään kertoo oman sijaintinsa ja oman pistoyksikön tiedonvarmistusryhmän sekä pyytää saada jatkossa tiedon kaikkien muiden topologien liikkeistä. OCM valvoo koko solmun olioita ja välittää niiden läsnäolomuutoksista tiedon kaikille niille olioille, jotka ovat asiaa kohtaa mielenkiintonsa ilmaisseet, joten kaikki solmun topologit saavat käynnistymisensä jälkeen tiedon muiden yksiköiden tiedonvarmistusjärjestelmien olemassaolosta ja yksiköiden varmistusryhmistä.

Saadessaan mahdollisesti käskyn irtautua solmutason toiminnasta topologi peruu rekisteröitymisensä OCM:lle, joten muut yksiköt saavat irti kytkeytymisestä tiedon välittömästi.





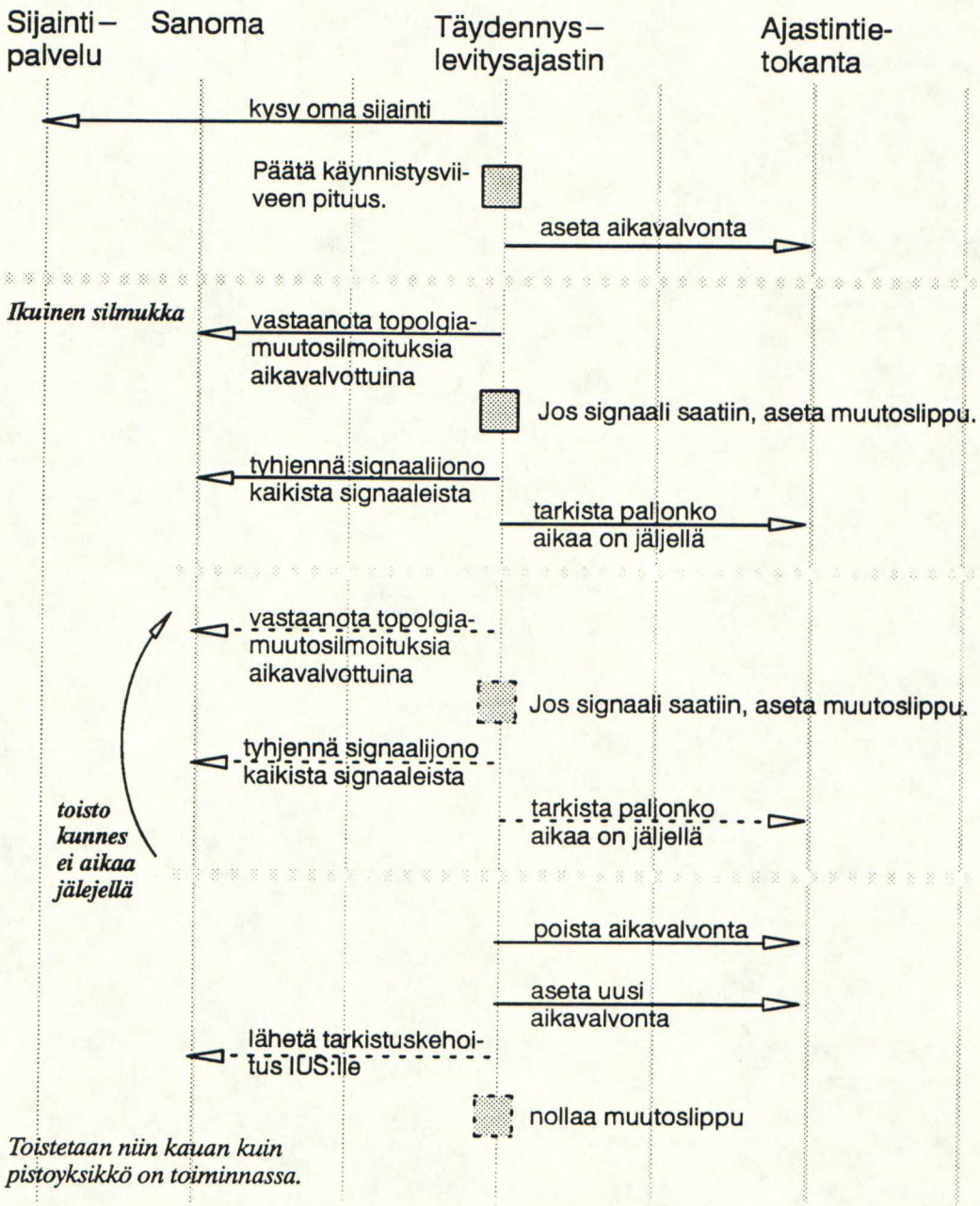
Ensimmäisen topologiapäätöksen käskytettyään topologi jää seuraamaan ympäristössä mahdollisesti tapahtuvia muutoksia ja toimii niiden mukaisesti:



Mikäli ei ole saatu irtikytkeytymiskäskyä, odotetaan seuraava sanoma.

Aikavalvonnan avulla toteutuu topologin toimintaa stabiloiva suodatus; OCM:n ilmoitukset välittyvät kuitenkin sanoma–ajurille ilman viivettä.

Lopuksi käsitellään täydennyslevitysajastimen toiminta, jolloin koko varmistusjärjestelmä on kuvattu ja määritelty. Ajastinoliossa on määritelty kaksi aikavakiota kuten kappaleessa 5.3.7 on määritelty; ensimmäistä arvoa käytetään käynnistyksessä ja toista siitä eteenpäin.



6 JOHTOPÄÄTÖKSET

SDH–solmuihin sovelletuista tiedonvarmistusmenetelmistä ei vielä ole saatavilla materiaalia, joten työhön ryhdyttiin puhtaalta pöydältä lähtien laitteiston ja ennalta määrätyn toteutusympäristön asettamista mahdollisuuksista ja rajoituksista.

Läpi koko työn tuli esille EEPROM:n ohjelmoimisen hitaus ja rajallinen uudelleenohjelmoitavuus, jotka johtavat varmistuspalvelun koon kasvuun. EEPROM:n hyvinä puolina vastaavasti korostuivat joustava yksittäisten tavujen käsittely, sillä palvelun käyttäjien varmistustarpeet vaihtelevat muutamista tavuista satoihin tavuihin, ja onnistuneen ohjelmoinnin jälkeinen riippumattomuus käyttöjännitteestä.

Sulautettu reaaliaikajärjestelmä sinänsä ei aiheuttanut ongelmia, vaan pikemminkin tarjosi mahdollisuuksia itsenäisiin ratkaisuihin, varsinkin kun laitteistoriippuvuudet ovat piilossa muun ohjelmiston takana.

Järjestelmälle asetettavien vaatimusten täsmentäminen vaati käyttäjien erilaisten tarpeiden vuoksi aikaa ja käytännön ratkaisusta muodostui kompromissi, jossa palvelun ja asiakkaan välinen rajapinta on yksinkertainen ja itse tiedonvarmistusjärjestelmä mahdollisimman riippumaton ohjelmiston muista osista.

Solmun pistoyksiköiden välisten suhteiden määritteleminen tiedostojen levittämisen mahdollistamiseksi vaikutti työn alkuvaiheessa suurimmalta haasteelta, mutta löytyneen solmutyypistä riippumattoman ratkaisun myötä painopiste siirtyi yksiköiden välisen yhteistyön suunnitteluun sellaiseksi, että mahdolliset lukkotilanteet vältetään. Valittu lähestymistapa, jossa yksiköt reagoivat toistensa lähettämiin signaaleihin erillisinä herätteinä muistamatta aiempia signaaleita, osoittautui hyväksi ratkaisuksi mahdollistaen järjestelmän toipumisen mihin tahansa solmun osaan kohdistuvasta häiriöstä.

Toteutuskieleksi määrätyn C++:n erityispiirteitä ei tässä ratkaisussa tarvita siinä määrin, etteikö vastaavaan tulokseen päästäisi millä tahansa proseduraalisella kielellä – mahdollisesti jopa järjestelmän suorituskyvyn parantuessa – mutta reaaliaikakäyttöjärjestelmä on ehdoton edellytys järjestelmän toiminnalle, koska tiedonvarmistusjärjestelmä esitettyjen rajoitteidensa vuoksi on selvästi hitaampi kuin useimmat sen palvelemat solmun ohjelmiston osat.

LÄHDELUETTELO

1. CCITT Recommendation G.707. Synchronous digital hierarchy bit rates. CCITT. 1992.

2. CCITT Recommendation G.708. Network node interface for the synchronous digital hierarchy. CCITT. 1992.

3. CCITT Recommendation G.709. Synchronous multiplexing structure. CCITT. 1992.

4. Saari, M. SDH–hallintaverkon hajautetun OSI–tietoliikenneohjelmiston sovellusohjelmointirajapinta. Kirkkonummi 1993. Lappeenrannan teknillinen korkeakoulu. 82 s.

5. Heinonen, M. Control Unit, requirement specification. Espoo 1991, Nokia Telecommunications. 10 s.

6. NTC Transmission Systems. Synfonet – Node equipment for synchronous digital hierarchy networks, product description. Espoo 1992, Nokia Telecommunications. 76 s.

7. Motorola. MC68302 integrated multi–protocol processor user's manual. Phoenix 1990, Motorola Inc. 312 s.

8. Lehto, A. Service Unit, requirement specification. Espoo 1991, Nokia Telecommunications. 13 s.

9. CCITT Recommendation X.701. Systems management overview for CCITT applications. CCITT. 1992.

10. Wood, P. SDH Application Software, requirement specification. Cambridge 1992, Nokia Telecommunications. 41 s.

11. Roberts, C. SDH Q3 Application Software. Nokia Telecommunications SDH Software Seminar, Espoo 22.9.1992. Cambridge 1992. Nokia Telecommunications. S. 1–16.

12. Raivola, M. NFL Software, technical specification. Espoo 1992, Nokia Telecommunications. 17 s.

13. Xicor. Data book. Kalifornia 1987, Xicor Inc. 685 s.

14. Hitachi. IC Memory no. 2 data book, EEPROM, flash memory, EPROM, OPTROM and mask ROM. 1992, Hitachi. 589 s.

15. Suihkonen, J. Memory Manager, behavioural specification. Espoo 1993, Nokia Telecommunications. 29 s.

16. Enea Data. OS68 reference manual. Täby 1992, Enea Data Ab. 496 s.

17. Kantelinen, P. An introduction to the design of concurrent processes. Nokia Telecommunications SDH Software Seminar, Espoo 8.3.1993. Espoo 1993. Laatukonsultointi. S. 1–39.