VTT TIEDOTTEITA – MEDDELANDEN – RESEARCH NOTES 1927

Kenttäväyläverkon automaattinen konfigurointi

Jarmo Lumpus
VTT Elektroniikka

VALTION TEKNILLINEN TUTKIMUSKESKUS
ESPOO 1998

ISBN 951–38–5336–5 (nid.)
ISSN 1235–0605 (nid.)

ISBN 951–38–5337–3 (URL: http://www.inf.vtt.fi/pdf/)
ISSN 1455–0865 (URL: http://www.inf.vtt.fi/pdf/)

JULKAISIJA – UTGIVARE – PUBLISHER

Valtion teknillinen tutkimuskeskus (VTT), Vuorimiehentie 5, PL 2000, 02044 VTT
puh. vaihde (09) 4561, faksi (09) 456 4374

Statens tekniska forskningscentral (VTT), Bergsmansvägen 5, PB 2000, 02044 VTT
tel. växel (09) 4561, fax (09) 456 4374

Technical Research Centre of Finland (VTT), Vuorimiehentie 5, P.O.Box 2000,
FIN–02044 VTT, Finland
phone internat. + 358 9 4561, fax + 358 9 456 4374

VTT Elektroniikka, Sulautetut ohjelmistot, Kaitoväylä 1, PL 1100, 90571 OULU
puh. vaihde (08) 551 2111, faksi (08) 551 2320

VTT Elektronik, Inbyggd programvara, Kaitoväylä 1, PB 1100, 90571 ULEÅBORG
tel. växel (08) 551 2111, fax (08) 551 2320

VTT Electronics, Embedded Software, Kaitoväylä 1, P.O.Box 1100, FIN–90571 OULU, Finland
phone internat. + 358 8 551 2111, fax + 358 8 551 2320

Toimitus Leena Ukskoski

Libella Painopalvelu Oy, Espoo 1998

Lumpus, Jarmo. Kenttäväyläverkon automaattinen konfigurointi [Automatic configuration of a fieldbus-based network]. Espoo 1998, Valtion teknillinen tutkimuskeskus, VTT Tiedotteita–Meddelanden–Research Notes 1927. 68 s. + liitt. 3 s.

Avainsanat LON, control systems, distributed systems, power distribution networks

TIIVISTELMÄ

Hajauttamisella pyritään lisäämään ohjaussovellusten modulaarisuutta ja joustavuutta, joka on tarpeen toimitettaessa voimakkaasti asiakaskohtaistettuja tuotteita ja järjestelmiä. Ohjaussovellusten hajautusratkaisujen perustana on usein kenttäväylätyyppinen verkkoratkaisu, mutta myös yleiskäyttöisemmät paikallisverkkoratkaisut, erityisesti Ethernet-verkot, ovat yleistymässä myös ohjaussovellusten hajautusratkaisujen alustoina. Hajautettujen sovellusten etuna keskitettyihin nähden on helpompi skaalattavuus ja johdotuksen ja sen rakentamiseen liittyvän työn kustannusten säästö.

Tässä työssä tutkittiin, kuinka hajautettujen järjestelmien konfigurointia voidaan tehostaa kenttäväylällä toteutetussa sähkönjakeluverkon suojaus- ja ohjausjärjestelmissä. Automatisoinnin toteutuksessa tulee ottaa huomioon nykyisten suunnittelu- ja konfigurointityökalujen palvelut ja rajoitukset. Erityisesti kiinnitettiin huomiota konfiguroinnin automatisoinnin toteuttavien päättelysääntöjen suunnitteluun ja toteutukseen. Päättelysäännöt sitovat verkon laitteet yhteen suunnittelijan määrämällä tavalla.

Konfiguroinnilla tarkoitetaan toimintojen lisäämistä ja poistamista sekä kommunikaation ja I/O-tietojen määrittelemistä verkkoon liittyneille laitteille. Konfiguroinnin avulla verkko saadaan toimimaan halutulla tavalla. Konfiguroinnin suorittaminen nykyisillä työkaluilla on työlästä ja virheherkkää. Yleisenä konfiguroinnin automatisoinnin vaikeutena on konfigurointien standardoinnin puute.

Ratkaisun ydin on verkkokonfiguraattorityökalu, joka suorittaa automaattista konfigurointia verkkosovellukseen. Verkkokonfiguraattori integroi projekti- ja laitetietokannat yhdeksi tietokannaksi ja tuottaa päättelyn avulla puuttuvat attribuutit verkon liityntöjen aikaan-saamiseksi. Konfiguroinnin automatisointi vaatii keskitetyn tiedonhallintajärjestelmän.

Työssä toteutettiin LON-verkon automaattisen konfiguroinnin toteuttava verkkokonfiguraattorityökalun prototyyppi, joka voidaan liittää osaksi suunnitteluprosessissa käytettäviä työkaluja. Prototyyppi suoritti konfiguroinnit oikein, ja työn tuloksena voidaan todeta, että konfigurointiproseduuria voidaan automatisoida ja näin vähentää kenttäväyläverkon suunnittelussa ja konfiguroinnissa tehtävää työtä.

Lumpus, Jarmo. Kenttäväyläverkon automaattinen konfigurointi [Automatic configuration of a fieldbus-based network]. Espoo 1998, Technical Research Centre of Finland, VTT Tiedotteita–Meddelanden–Research Notes 1927. 68 p. + app. 3 p.

Keywords LON, control systems, distributed systems, power distribution networks

ABSTRACT

The aim of distribution in control systems is to increase modularity and flexibility, which is needed when supplying highly customized products and systems. Distributed control systems are usually based on fieldbus-based network solutions, but also LAN-based networks (especially Ethernet) have been used as distribution platform. Advantages of distributed systems as compared to centralized systems include easier scalability and cost savings in wiring and construction.

This thesis investigates ways to improve the configuration of distributed systems in case of a fieldbus-based protection and control system for power distribution networks. When implementing the automation, the restrictions and services of existing design and configuration tools must be taken into account. Special attention must be paid to the design and implementation of reasoning rules which perform the automatic configuration. The reasoning rules interconnect the network's devices according to the designer's specifications.

Configuration means the addition and removal of functions, as well as the specification of communication and I/O features of the network's devices. The desired network behavior is achieved by means of configuration. Performing configuration tasks using current tools is a laborious and error prone task. Generally speaking the main difficulty of automatic configuration is the lack of configuration standards.

The core of our solution is a network configurator tool which configures applications automatically. The network configurator integrates project and device data-bases into one data-base and produces the missing attributes by means of the reasoning rules to interconnect the devices in the network. The automatic configuration requires a centralized data management system.

In this thesis a prototype of the network configurator was implemented which carried out the automatic configuration of a LON network. The network configurator can be added to a set of tools used in the design process. The prototype performed the configuration correctly and we can conclude that the automation of the configuration process is feasible and decreases the work in design and configuration of a fieldbus-based network.

ALKULAUSE

Tämä työ liittyy VTT Elektroniikassa tehtyyn DYNAMO-tutkimusprojektiin, jossa kehitettiin hajautettujen järjestelmien ohjelmistojen adaptiivisuutta ja konfigurointia tukevia ratkaisuja. Työn tuloksia sovellettiin ABB Transmit Oy:n tarjoamaan pilottiin. Työn tarkoituksena oli selvittää, miten kenttäväyläverkon automaattinen konfigurointi tulisi suorittaa.

Työn valvojana on toiminut Oulun yliopistosta apulaisprofessori Juha Röning, jolle esitän kiitokseni työni valvonnasta ja tarkastamisesta. Kiitokset myös toiselle tarkastajalle professori Olli Silvénille.

Kiitän myös työn ohjaajaa fil. maist. Eila Niemelää ja tekn. lis. Harri Perunkaa asiantuntevista kommenteista sekä dipl.ins. Jouni Heikkistä pilotin sovellusanalyysin suorittamisesta.

Kiitokset työkavereille.

Oulussa 22.5.1998

Jarmo Lumpus

SISÄLLYSLUETTELO

LYHENTEIDEN JA MERKKIEN SELITYKSET

CAN Controller Area Network. Kenttäväylä.

CAP Computer Aided Programming. Tietokoneavustettu ohjelmointi.

CEN Comité Européen de Normalisation. Standardoimisjärjestö.

CENELEC Comité Européen de Normalisation Electrotechnique. Standardoimisjärjestön sähkötekniikan alajaosto.

CLI Call Level Interface. Kutsurajapinta tietokantaan.

CRC Cyclic Redundancy Check. Virheenkorjausalgoritmi.

CSMA Carrier Sense Multiple Access. Väylänvalvonta-algoritmi.

DDL Device Description Language. Laitekuvauskieli.

DLL Dynamic Link Library. Dynaamisesti linkitetty kirjasto.

DP Decentralized Peripherals. Profibusin kenttäväylä (Profibus DP).

FBD Function Block Diagram. IEC 1311 -standardin ohjelmointikieli.

FF Foundation Fieldbus. Kenttäväylä.

FIP Factory Instrumentation Protocol. Kenttäväylä.

IEC International Electro-technical Commision. Standardoimisjärjestö.

IL Instruction List. IEC 1311 -standardin ohjelmointikieli.

ISA Instrument of Society American. Standardoimisjärjestö.

ISO International Organization of Standardization. Standardoimisjärjestö.

LAN Local Area Network. Paikallisverkko.

LD Ladder Diagram. IEC 1311 -standardin ohjelmointikieli.

LNT Lon Network Tool. LON-väylän konfigurointityökalu.

LON Local Area Network. Kenttäväylä.

LSG Lon/Spacom Gateway. Yhdyskäytävä.

MAC Media Access Control. Neuronpiirin prosessori.

MFC Microsoft Foundation Classes. Microsoftin luokkakirjasto.

MSC Message Sequence Chart. Viestien järjestyksen kuvaava kaavio.

NC Network Configurator. Verkkokonfiguraattori.

NVCT Network Variable Configuration Table. Verkkomuuttujien konfigurointitaulu neuronpiirissä.

ODBC Open Database Connectivity. Microsoftin määrittelemä kutsurajapinta tietokantaan.

OSI Open Systems Interconnection. Malli, johon useat protokollat perustuvat.

PLC Programmable Logic Controller. Ohjelmoitava logiikka.

RECAP RElay Configuration and Programming. Releiden konfigurointi- ja suunnittelutyökalu.

REF RElay Feeder terminal. Relelaitetyyppi.

RES RElay Signal alarm device. Relelaitetyyppi.

SA/SD Structured Analysis/Structured Design. Rakenteellinen ohjelmiston kuvaus.

SCPT Standard Configuration Parameter Type. LON-protokollassa käytetty standardisoitu konfigurointiparametrityyppi.

SDS Smart Distributed System. CAN-väylän protokolla.

SFC Sequential Function Chart. IEC 1311 -standardin ohjelmointikieli.

SI/SD Self Identification/Self Documentation. LON-väylän laitteilta voidaan lukea sen tietoja.

SNVT Standard Network Variable Type. LON-protokollassa käytetty standardisoitu verkkomuuttujatyyppi.

ST Structured Text. IEC 1311 -standardin ohjelmointikieli.

TCP Transport Control Protocol. TCP/IP:n protokolla.

TCP/IP Transmission Control Protocol/Internet Protocol. Protokollaperhe.

UDP User Datagram Protocol. TCP/IP:n protokolla.

1. JOHDANTO

Viime vuosina ohjaussovellusten hajauttaminen on yleistynyt. Hajauttamisella pyritään lisäämään ohjaussovellusten modulaarisuutta ja joustavuutta, joka on tarpeen toimitettaessa voimakkaasti asiakaskohtaistettuja tuotteita ja järjestelmiä. Ohjaussovellusten hajautusratkaisujen perustana on usein kenttäväylätyyppinen verkkoratkaisu, mutta myös yleiskäyttöisemmät paikallis-verkkoratkaisut, erityisesti Ethernet-verkot, ovat yleistymässä myös ohjaussovellusten hajautusratkaisujen alustoina. Kaupallisesti saatavat CAN- ja LON-pohjaiset kenttäväylät ovat yleisiä hyvää toimintavarmuutta ja kustannustehokkuutta vaativissa ohjaussovelluksissa, kuten auto- ja koneteollisuudessa sekä kiinteistäautomaatioon ja sähkönjakeluun liittyvissä sovelluksissa. Profibus ja Interbus ovat taas yleisiä teollisuusautomaation sovelluksissa, joissa korkea toimintavarmuus ja laatu merkitsevät vielä enemmän kuin kustannustehokkuus.

Koska kaupallisia hajautusratkaisuja on jo runsaasti saatavilla, ei merkittävää kilpailuetua siksi ole saavutettavissa pelkästään kenttäväylää tai muuta vastaavaa hajautusratkaisua hyödyntämällä. Merkittävän kilpailuedun yritys voi sen sijaan saavuttaa, jos sen tuotteisto on helposti ja nopeasti konfiguroitavissa erilaisiin asiakastarpeisiin siten, että hajautetut ohjaussovelluksetkin voidaan toimittaa sovitussa aikataulussa ja laadukkaasti.

Ongelmana hajautetuissa ympäristöissä on sovellusten suunnittelun monimutkaisuus ja ainutkertaisuus. Jokainen toimitus on ainakin joiltakin osin erilainen, vaikka tuotteistossa olisikin selvästi tunnistettava muuttumaton ydin. Järjestelmätoimittajan kannalta tämä merkitsee sitä, että jokainen toimitus on suunniteltava erikseen. Sovellusten suunnittelu ja ylläpito edellyttävät perusteellista teknistä asiantuntemusta. Epäyhtenäisistä ja suunnittelijakohtaisista käytännöistä johtuen suunnittelutietojen uudelleenkäyttöaste on alhainen ja ylläpito työlästä. Tämän vuoksi on tarve kehittää tehokkaita järjestelmän suunnittelu- ja konfigurointimenetelmiä, joissa suunnittelutyön osuutta voidaan pienentää uudelleenkäyttöä ja suunnitteluautomaatiota lisäämällä.

Tässä työssä tarkasteltiin suunnittelutiedon uudelleenkäytön tehostamista hajautettujen ohjaussovellusten tapauksessa. Tarkastelussa mielenkiinto kohdistetaan valmiiden ohjelmistokomponenttien uudelleenkäyttöön, konfiguroimiseen ja allokointiin kenttäväylällä hajautetussa toimintaympäristössä. Työn tarkoituksena on kehittää ja kokeilla keinoja, joilla suunnittelijoiden työtä voidaan yhdenmukaistaa ja tehostaa.

Työn tuloksena esitetään yrityksen tarpeisiin mukautettu verkkokonfiguraattorityökalun prototyyppi, jossa hyödynnetään sovellusaluekohtaisia päättelysääntöjä verkon ja sovelluskomponenttien konfiguroimiseksi. Verkkokonfiguraattorin tehtävänä on päättelysääntöjen mukaisesti kytkeä hajautetun järjestelmän alijärjestelmien tulot ja lähdöt toisiinsa mahdollisimman automaattisesti. Toteutuksessa huomioidaan nykyisten suunnittelu- ja konfigurointityökalujen tarjoamat palvelut ja rajoitukset.

2. OHJAUSSOVELLUSTEN HAJAUTUSRATKAISUT

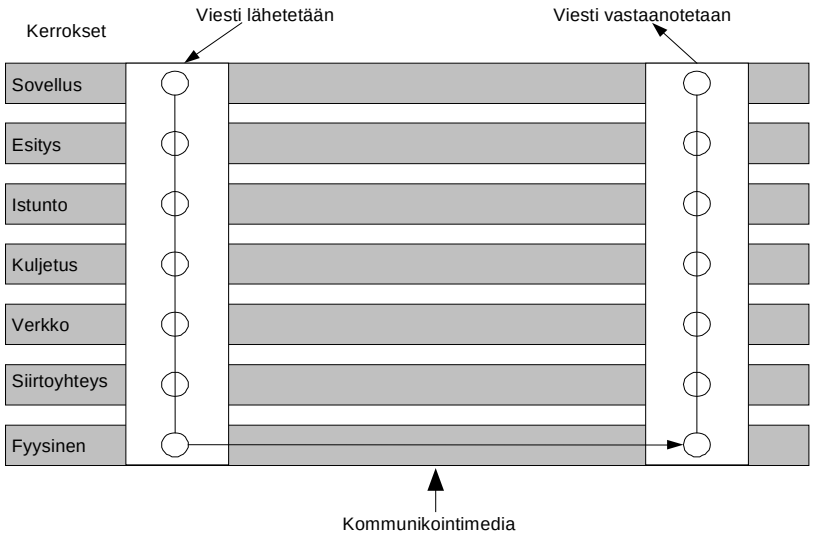
2.1 Yleistä hajautetuista järjestelmistä

2.1.1 Hajautetut järjestelmät ja protokollat

Hajautetulla järjestelmällä tarkoitetaan tietokoneiden verkkoa, jossa verkon solmuissa sijaitsevat tietokoneet on kytketty toisiinsa mahdollisimman läpinäkyvän verkon avulla siten, että solmuissa sijaitsevat sovellukset voivat kommunikoida keskenään. Jos verkko on toteutettu läpinäkyväksi, sovellukset voivat kommunikoida täysin riippumatta niiden sijainnista.

Jotta sovellukset voisivat kommunikoida, tarvitaan fyysisen verkon lisäksi solmujen tunnisteet ja kommunikointisäännöt eli verkon noudattama protokolla. Protokollia on useita riippuen niiden käyttötarkoituksesta. Protokollille on ominaista se, että ne pyritään standardoimaan tai toteuttamaan jo olemassa oleviin standardeihin perustuen. ISO:n (International Organization of Standardization) OSI-malli (Open Systems Interconnection) on yleisin perusta, jolle protokollat on toteutettu.

OSI-malli ei ota kantaa järjestelmien sisäiseen rakenteeseen vaan siihen, miltä ne näyttävät tietoliikenteen kannalta. Malli jakautuu seitsemään eri loogiseen kerrokseen, joille kullekin on määritelty omat tehtävät ja palvelut. Palveluita tarjotaan ylemmälle kerrokselle näkymättömän rajapinnan kautta. Kuvassa 1 on OSI-mallin kerrokset [1, s. 69–71].

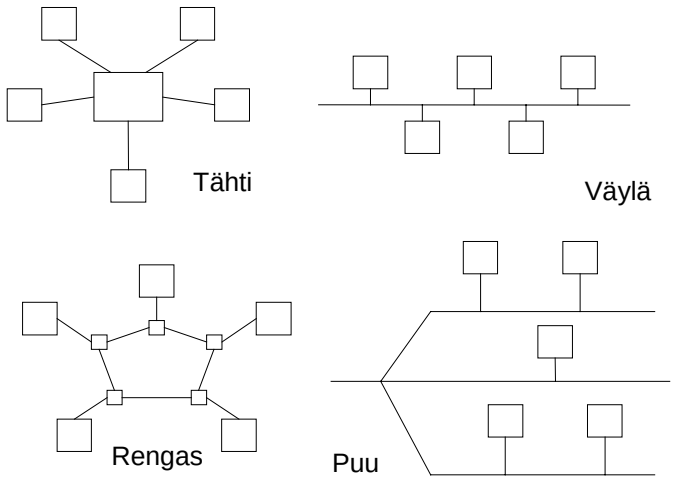


Kuva 1. OSI-mallin kerrokset ja viestin kulku kerrosten läpi.

TCP/IP-protokollaperhe (Transmission Control Protocol/Internet Protocol) on kehitetty Internetiä ja muita yhdistettyjä verkkoja varten. Protokolla on kerrostettu mutta ei noudata täydellisesti seitsemänkerroksista OSI-mallia. TCP/IP:n käyttämät kerrokset ovat sovellus-, kuljetus-, Internet-, verkkoliityntä- ja alla oleva verkkokerros. Internet-kerroksessa on mahdollisuus käyttää kahta eri protokollaa, joita alempana oleva IP-verkkoliityntäkerros tukee. Nämä protokollat ovat TCP (Transport Control Protocol) ja UDP (User Datagram Protocol). TCP on luotettava, kun taas UDP ei takaa luotettavaa tiedonsiirtoa. TCP/IP:n suosio perustuu siihen, että se ei ole riippuvainen siirtoteknologiasta, jolloin voidaan yhdistää hyvinkin erilaisia verkkoja toisiinsa. TCP:tä käytetään useissa Internet-palveluissa, kuten FTP:ssä ja Telnetissä [1, s. 84–85].

2.1.2 Hajautustopologiat ja verkon käytön hallinta

Kuvassa 2 on neljä erilaista hajautustopologiaa. Tässä työssä keskityttiin erityisesti väyläpohjaisiin topologioihin. Väylä on puutopologian erikoistapaus, jossa on vain yksi runko eikä yhtään haaraa. Väylä- ja puutopologiat ovat käytetyimpiä paikallisverkkojen hajautusratkaisuissa. Väylätopologiassa kaikki laitteet jakavat saman fyysisen verkon, mutta vain yksi laite saa käyttää väylää kerrallaan. Normaalitilassa laite lähettää paketin, jossa on vastaanottajan osoite. Paketti menee kaikille verkon laitteille, mutta ainoastaan vastaanottajaksi osoitettu laite kopioi paketin itselleen [5, s. 329–330].



Kuva 2. *Hajautustopologiat.*

Koska kaikki laitteet voivat haluta lähettää viestejä toisilleen samanaikaisesti, tarvitaan protokolla, jolla hallitaan verkon käyttöä ja mahdollisia yhteentörmäyksiä

samanaikaisten lähetysten varalta. Suosituin verkon samanaikaista käyttöä valvova menetelmä puu- ja väylätopologiassa on CSMA (Carrier Sense Multiple Access), mutta myös tokeniin perustuvia verkonhallintakäytäntöjä on toteutettu.

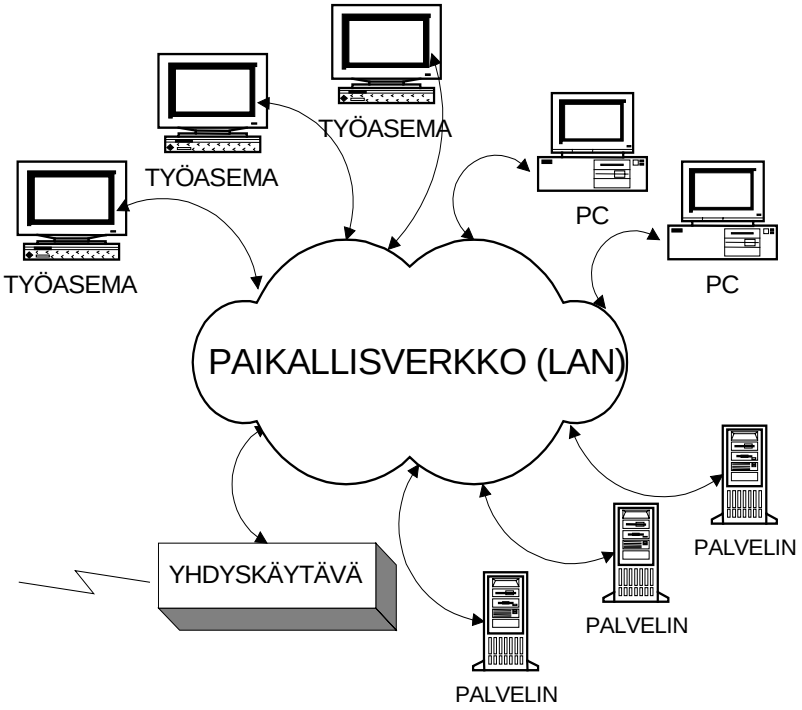
Kun CSMA:ta noudattavassa verkonhallinnassa havaitaan pakettien törmäys, kaikille solmuille lähetetään viesti törmäyksestä. Sen jälkeen jokainen solmu, joka haluaa lähettää paketin, odottaa satunnaisen ajan ja yrittää lähettää paketin uudelleen. CSMA-periaate toimii hyvin, kun verkon kuormitus on noin 70 % nimelliskapasiteetista. Tätä suuremmilla kuormituksilla yhteentörmäykset lisääntyvät ja verkon läpäisykyky laskee selvästi. Toinen verkon käytön hallintaan kehitetty väylänhallintamekanismi perustuu tokenien käyttöön. Periaatteena on se, että verkossa kiertää koko ajan token solmusta solmuun. Se solmu, jolla on token hallussaan, saa verkon aina haltuunsa. Kun solmu on käyttänyt verkkoa, sen on lähetettävä token eteenpäin seuraavalle solmulle. Token bus -verkot ovat suhteellisen deterministisiä ja siksi ne soveltuvat CSMA-periaatetta hyödyntäviä paremmin turvallisuuskriittisiin sovelluksiin. Ainoa merkittävä vikaantumislähde on tokenin häviäminen, jota varten on kehitetty algoritmeja uuden tokenin luomiseksi verkkoon [5, s. 297–299].

2.2 Hajautetut järjestelmät käytännössä

2.2.1 Toimistoautomaation sovellukset (paikallisverkot)

Kuva 3 esittää paikallisverkon (LAN, Local Area Network) avulla toteutettua hajautettua toimistoautomaatiojärjestelmää, jossa työasemat, PC:t, serverit ja kirjoittimet on liitetty toisiinsa. Kuvan 3 mukaisia järjestelmiä käytetään hyvin paljon yrityksissä, kouluissa ja toimistoissa. Toimistoautomaatiosovellukset eivät ole aikakriittisiä, ja yleensä ihmisten turvallisuus ei ole riippuvainen järjestelmän toimivuudesta. Toiminnallisuutta pystytään lisäämään vikasietoisuuden kustannuksella [1, s. 2].

Tiedonsiirtonopeus LAN-verkossa voi olla jopa 20 Mbit/s ja etäisyys verkon solmuilla 25 kilometriä. LAN on erittäin käyttökelpoinen, kun verkkoon on liittynyt hyvin erilaisia laitteita ja tiedonsiirtotavat laitteiden välillä ovat kirjavia [5, s. 330–331].



Kuva 3. Hajautettu toimistoautomaatiojärjestelmä.

2.2.2 Ohjaussovellusten hajautusratkaisut

Teollisuusautomaatiossa, olipa sitten kyse prosessiautomaatiojärjestelmistä tai kone- ja laiteautomaation sovelluksista, tarvitaan yhä enemmän hajautettua ohjausta. Keskitetyistä järjestelmistä ollaan siirtymässä halpoihin mikrokontrollereihin ja mikroprosessoreihin perustuviin moduuleihin. Jokainen moduuli keskittyy tekemään ainoastaan omaa, sille ohjelmoitua ja konfiguroitua tehtäväänsä. Lisäksi moduulit voivat kommunikoida keskenään sarjamuotoisesti esim. kenttäväylän avulla.

Hajautetuille ohjaussovelluksille ominainen piirre on lyhyiden viestien käyttö tiedonsiirrossa sekä lyhyet ja ennustettavissa olevat vasteajat [22]. Kuvassa 4 on eräs hajautettu ohjaussovellus, jota käytetään teollisuuden sähkönjakelusovelluksissa.

Hajautetuissa järjestelmissä käytetyt tiedonsiirtoväylät voidaan jakaa seuraaviin luokkiin:

luokka 0: anturi- ja toimilaiteväylät (I/O-väylät),

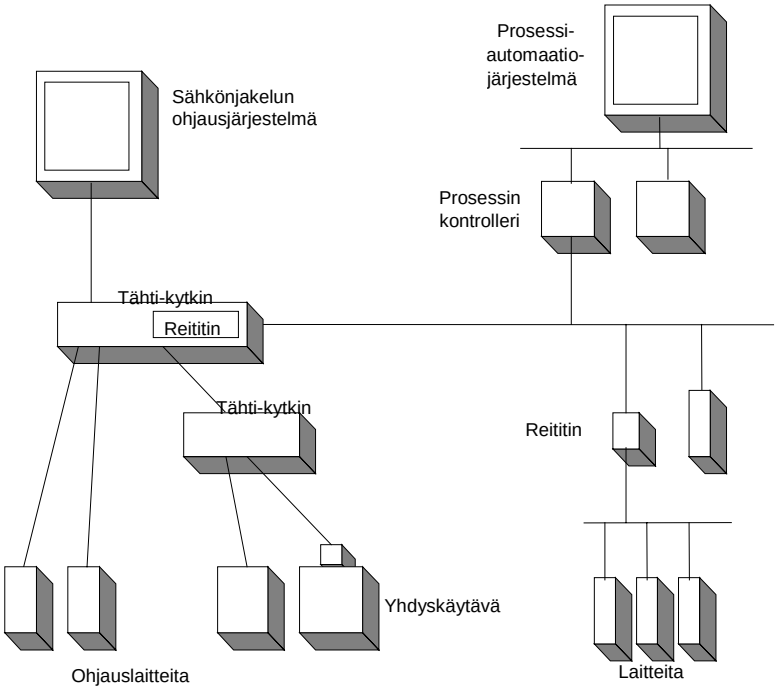
luokka 1: varsinaiset kenttäväylät,

luokka 2: tehdasväylät,

luokka 3: paikallisverkot (LAN) ja

luokka 4: suuralueverkot (WAN).

Automaation tiedonsiirtotarpeet tyydytetään luokilla 0, 1 ja 2, mutta suuntaus on yhä enemmän siihen, että käytetään ylemmän tason tiedonsiirtoa myös ohjaussovelluksissa [31]. Nykypäivänä automaatiosovelluksissa käytetään jopa TCP/IP-protokollaa ja tiedonsiirtoväylänä LAN-verkkoa. Tämän ovat mahdollistaneet koko ajan kehittyvät tietoliikenneratkaisut ja niiden kaupallistuminen.



Kuva 4. Hajautettu ohjaussovellus.

2.2.3 Toimistoautomaation ja ohjaussovellusten hajautusratkaisujen erot

Toimistoautomaatiossa hajautus eroaa monella tapaa siitä hajautuksesta, mitä käytetään esim. automaatioteollisuudessa. Eroa kuvaa esim. suorituskyvyn mittaaminen. Toimistoautomaatiossa suorituskykyä mitataan yleensä yksiköllä bittiä sekunnissa, kun taas teollisuussovellusten puolella suorituskykyä mitataan enemmänkin viestejä sekunnissa -yksiköllä [1, s. 30–31].

Teollisuuden tiedonsiirtotarpeet ovat hyvin erilaiset kuin toimistoympäristön tarpeet. Teollisuusprosessit toimivat yötä päivää, eikä katkoksia sallita. Katkokset voivat aiheuttaa suuria kustannuksia tai vaaratilanteita. Myös erikoiset olosuhteet aiheuttavat omat ongelmansa tiedonsiirrolle. Teollisuussovelluksessa on yleensä paljon enemmän laitteita kuin toimiston tietojärjestelmässä. Tehtaasta riippuen laitteita voi olla jopa kymmeniä tuhansia. Laitteet ovat kuitenkin paljon yksinkertaisempia kuin toimistoautomaatiossa. [3]

Ohjaussovelluksen solmujen välimatkat ovat kertaluokkaa pienemmät kuin paikallisverkoissa, sillä paikallisverkossa laitteet voivat sijaita useiden kilometrien päässä toisistaan. Ohjaussovellusten aikakriittisyyden vuoksi etäisyydet ovat yleensä alle kilometrin. Reitittimien avulla välimatkaa saadaan pidennettyä, mutta tällöin sovelluksen toiminta hidastuu.

2.2.4 Hajautuksen edut ja haitat ohjausjärjestelmissä

Hajautetulla järjestelmällä on monia etuuksia keskitettyyn järjestelmään nähden. Hajautettuja sovelluksia on helpompi skaalata kuin keskitettyjä järjestelmiä. Kapasiteetin kasvattaminen hajautetuissa ohjausjärjestelmissä on mahdollista lisäämällä uusi solmu järjestelmään. Toisaalta myös toiminnallisuutta voidaan lisätä lisäämällä uusia toimintoja järjestelmässä jo oleviin solmuihin tai lisäämällä sekä solmu että toiminto. Keskitetyssä järjestelmässä toiminnallisuuden lisääminen on mahdollista vain, jos olemassa oleva kapasiteetti sen sallii.

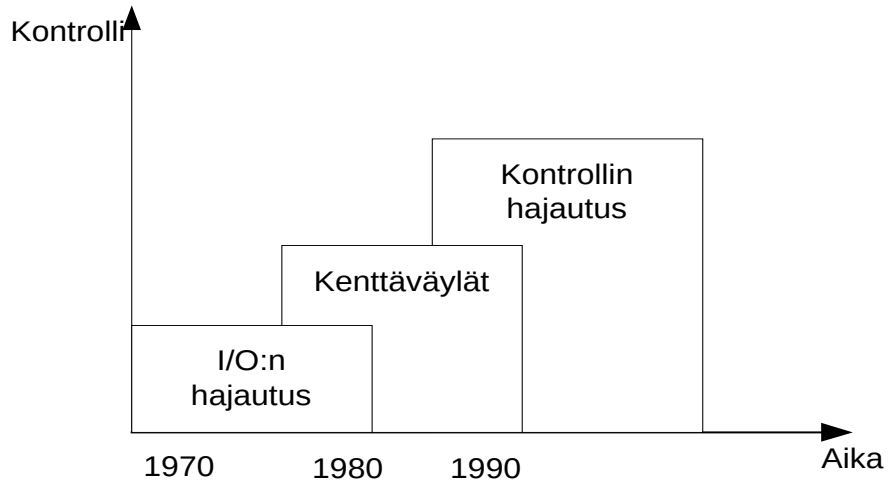
Hajautetussa ohjausjärjestelmässä voidaan säästää kustannuksia johdotuksen ja sen rakentamiseen liittyvän työn osalta. Keskitetyssä järjestelmässä tarvitaan erillinen johdotus jokaiselle laitteelle, kun taas hajautetussa järjestelmässä laitteet pystyvät kommunikoimaan yhteisellä medialla esim. väylän avulla. Analoginen isäntä-orja-systeemi tarvitsee kaksinkertaisen kaistanleveyden digitaaliseen kohteesta kohteeseen kommunikointiin verrattuna, kun käytetään negatiivinen hyväksyminen -vastauspalvelua (unacknowledge). Tilanne pahenee, jos yhden laitteen sisällä on silmukoita. Silmukoita syntyy, jos laitteessa on sekä anturi että anturin antamaa tietoa muuntava yksikkö (transducer) [4].

Yleensä järjestelmien monimutkaisuus pyrkii kasvamaan hajautuksen tuomien mahdollisuuksien myötä. Verkon hallinnan kanssa voi tulla ongelmia ja järjestelmän kokonaissuorituskyky voi jopa laskea, mikäli suunnittelu tehdään huolimattomasti. Lisäksi jos väylä menee fyysisesti poikki, niin osa verkon solmuista on tämän jälkeen toimintakyvyttömiä. Kenttäväyliä käytetään hyvin paljon työkoneissa, joissa väylän fyysinen katkeaminen on hyvinkin todennäköinen. Analogisissa isäntä-orja-järjestelmissä siirtojärjestelmän fyysinen katkeaminen ei aiheuta yleensä kuin yhden laitteen toimintakyvyttömyyden.

2.3 Ohjaussovellusten hajautustasot

2.3.1 Yleistä

Kuvassa 5 esitetään eri hajautusmahdollisuuksien käyttöä ja kehitystyötä eri aikakausina. 1960- ja 1970-luvuilla ohjaussovellusten hajauttamisessa keskityttiin I/O:n hajautta-miseen. Digitaalisen kenttäväylän kehittäminen ja standardointi aloitettiin 1980-luvulla ja kehitystyö jatkuu edelleen. 1990-luvulla on kehitetty ohjauksen hajautusta tukevia ratkaisuja automaatiosovelluksissa. Kuvassa eri tyyppien käyttö ei ole suinkaan loppunut silloin, kun kuva ehkä antaa ymmärtää, vaan tarkoitus on kuvata sitä, milloin mikin tyyppi on ollut alalla vallitseva ja kehitystyö aktiivisimmillaan. Myös I/O:n hajautusta käytetään edelleen, mutta suurin osa tiedonsiirrosta hoidetaan nykyään kenttäväylillä. Ohjauksen hajautus on enemmänkin tulevaisuudessa käytettävä hajautustapa, vaikka siitäkin on jo käytännön sovelluksia. Suuntaus on kuitenkin koko ajan siirtyä ylemmälle tasolle toiminnallisuuden suhteen.



Kuva 5. Hajautustasot teollisuusautomaatiossa.

2.3.2 I/O:n hajauttaminen

I/O:n hajautusta käytetään mittaus- ja tilatietojen sekä toimilaiteohjauksiin liittyvien viestien siirtoon. Aikakriittisen luonteen vuoksi I/O:n hajautuksessa pyritään mahdollisimman pieniin vasteisiin ja pieneen viestien kokoon. Viestien koot on yleensä ilmoitettu bitteinä ja vasteajat millisekunteina.

Esimerkkinä I/O-hajautuksen mahdollistavasta ratkaisusta on Profibus DP (Decentralized Peripherals). Se on nopea (1,5–12 Mb/s) väyläratkaisu, joka on tarkoitettu ennen kaikkea aikakriittisiin automaatiosovelluksiin [7]. Profibus DP:ssä käytetään melkein aina siirtojärjestelmänä RS 485 -liitäntäistä kierrettyä parikaapelia. Järjestelmässä keskusohjaimet (PLC:t ja PC:t) kommunikoivat hajautettujen kenttälaitteiden kanssa erittäin nopean sarjamuotoisen siirtotien avulla. Keskusohjain keskustelee kenttälaitteiden kanssa syklisesti eli lukee vuorotellen jokaiselta orjalta tulo-informaatiota. Syklin tulee olla pienempi kuin ohjelman syklin nopeus keskus-PLC:ssä. Maksimi solmujen lukumäärä Profibus DP:ssä on 126 (yhdellä väylällä). Väylällä voidaan käyttää joko yhtä tai useampaa keskusohjainta (master) [20].

2.3.3 Toiminnan hajauttaminen

Toiminnallisessa hajautuksessa verkon solmuissa on enemmän älykkyyttä kuin perinteisissä I/O-laitteissa. Jokainen solmu suorittaa jotain sille määrättyä tehtävää, esimerkiksi liikkeen tunnistusta. Koko verkolla voidaan suorittaa hyvin monimutkaisia ohjaussovelluksia. Toiminnallinen hajautus tehdään käytännössä pääasiassa kenttäväylien avulla.

Esimerkkinä toiminnallisesta hajautuksesta on LON-kenttäväylä, jonka suurin tiedonsiirtonopeus on 1,25 Mbit/s (I/O:n hajautuksessa Profibus DP:n nopeus on 12 Mbit/s). LON-väylän laitteet sisältävät älykkyyttä (mikroprosessori), joka toteuttaa protokollan ja suorittaa ohjaustehtäviä.

2.3.4 Ohjauksen hajauttaminen

Kun puhutaan ohjauksen hajautuksesta, tarkoitetaan yleensä koko järjestelmän kattavaa älykästä hajautettua ohjausta. Ohjauksen hajauttaminen on hyvin vaativa tehtävä, koska eri puolilla järjestelmää toimivien osaohjauskeskuksien tulee kyetä toimimaan yhteistyössä muiden osaohjauskeskusten kanssa. Muissa solmuissa tapahtuvat poikkeustilanteet ja niiden vaikutusten hallinta on huomattavasti monimutkaisempaa hajautetun ohjauksen tapauksessa kuin keskitetyssä ohjauksessa, jossa päätökset tapahtuvat aina samassa järjestyksessä ja samanaikaisesti.

Erilaisia tekoälyyn perustuvia tekniikoita on käytetty hajautetun ohjauksen toteuttamiseen. Viime aikoina agenttitekniikoilla on yritetty ratkaista hajautetun ohjauksen ongelmia. Agenteille on olemassa kaksi määritelmää, vahva ja heikko. Heikolla määritelmällä ilmaistaan laitteisto- tai ohjelmistopohjaista järjestelmää, jolla on neljä ominaisuutta. Taulukossa 1 on selitetty nämä ominaisuudet.

Taulukko 1. Heikon agenttimääritelmän ominaisuudet.

Ominaisuus	Selitys
Autonomisuus	Agentit toimivat ilman, että ihminen puuttuu niiden toimintaan tai sisäisiin tiloihin.
Yhteistyökyky	Agentit kommunikoivat keskenään jollain agentti-kommunikaatiokielellä.
Reaktiivisuus	Agentit havaitsevat ympäristöään ja vastaavat siinä tapahtuviin muutoksiin.
Ennakoiva käyttäytyminen	Agentit eivät pelkästään vastaa ympäristön herätteisiin vaan ne voivat tehdä omia aloitteita päästäkseen tiettyyn päämäärään.

Vahvaa määritelmää suositaan varsinkin tekoälytutkimuksen parissa. Vahvassa määritelmässä agentilla tarkoitetaan tietokonejärjestelmää, joka sisältää yllä olevien heikon agentin ominaisuuksien lisäksi myös toiminnallisuutta, joka yhdistetään ihmisen käyttäytymiseen. Eräät tutkijat ovat kehitelleet jopa ihmisen aisteja matkivia agenttiratkaisuja. [32]

Agenttiarkkitehtuurit

Agenttiarkkitehtuureja on useita ja tässä työssä esitellään niistä seuraavat neljä:

- älykkäät agentit
- reaktiiviset agentit
- hybridiagentit ja
- liikkuvat agentit.

Älykkäitä agentteja käytetään erittäin paljon tekoälysovelluksissa ja roboteissa. Älykkäitä agentteja käytetään erityisesti sellaisissa sovelluksissa, missä tarvitaan korkean tason päättelyä. Älykkäät agentit eivät välttämättä sovi hajautettuihin ohjainsovelluksiin, koska ne ovat melko hitaita ja yleensä monimutkainen päättely ei ole tarpeen ohjainsovelluksissa.

Reaktiiviset agentit eivät hallitse ympäristönsä sisäisiä ja symbolisia malleja, vaan ne vastaavat herätteisiin siinä ympäristössä, mihin ne ovat sulautettu. Hybridiajatuksessa yhdistetään harkitsevien (deliberative) ja reaktiivisten agenttien toiminta-ajatukset.

Hybridi- ja reaktiiviset agentit soveltuvat ohjainsovelluksiin, koska niissä on staattinen kontrollin hajautus.

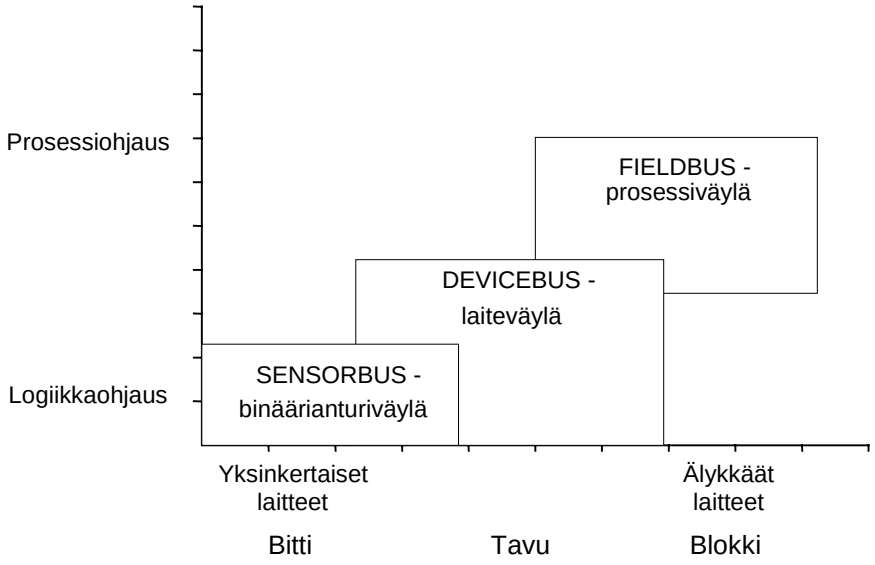
Liikkuva agentti on ohjelma, joka voi vaihtaa paikkaa ja silti säilyttää tilansa. Liikkuvan agentin kohdalla voidaan puhua dynaamisesta kontrollin hajautuksesta. Sovelluksia liikkuvista agenteista ohjainsovelluksissa on erittäin vähän. Ne soveltuvat enemmänkin tiedonkeruuseen ja valvontatehtäviin, jossa päättely tehdään paikallisesti. [25, 32]

2.4 Kenttäväylät

2.4.1 Yleistä

Analogiatekniikka on ollut teollisuuden tiedonsiirtojärjestelmien perustana 1960-luvulta lähtien. Silloin käytettiin 4–20 mA:iin perustuvaa tiedonsiirtotapaa, jossa jokaisesta kenttälaitteesta oli parikaapeliyhteys valvomoon. Etuina järjestelmässä olivat mm. avoimuus, nopeat vasteet ja kommunikaation yksinkertaisuus.

Tiedonsiirtotarpeiden kasvaessa ja vaatimusten kohotessa on siirrytty yhä enemmän digitaalitekniikkaan. Digitaalitekniikka mahdollistaa luotettavamman informaation välityksen ja monipuolisemman kommunikaation laitteiden välillä. Kehityksen seurauksena on syntynyt tehokkaampi tiedonsiirtotekniikka, kenttäväylä. Kenttäväylä on digitaalinen, kaksisuuntainen ja väyläpohjainen tiedonsiirtojärjestelmä, joka yhdistää älykkäät kenttälaitteet ja muun automaation koko tehtaan kattavaksi tietoverkoksi [3].



Kuva 6. Väylätyypit.

Kuvassa 6 on esitetty väylätyypit ja niiden käyttötarkoitukset. Kenttäväylissä käytetään älykkäitä laitteita ja viestien koko on suurempi kuin laite- ja anturiväylillä.

Kenttäväylien älykkyys mahdollistaa korkeamman tason toiminnallisuuden kuin kuvan 6 muut väylätyypit.

Vaikka kenttäväyläpohjainen ratkaisu ei olekaan keskitetty järjestelmä, siinä voi silti olla isäntä- ja orjasolmuja. Tässä tapauksessa hajautuksen edut menetetään osittain, varsinkin jos on vain yksi isäntä. Teollisuusautomaatiossa suosittu ratkaisu on monen isännän järjestelmä. Jotta hajautuksesta saataisiin mahdollisimman paljon irti, täytyy käyttää kohteesta kohteeseen -arkkitehtuuria.

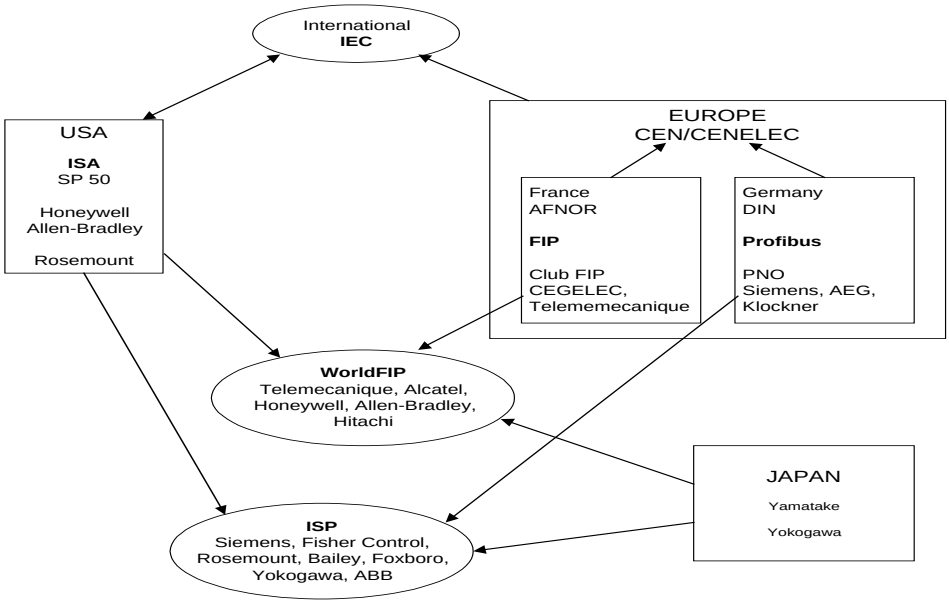
Avoimet kenttäväyläratkaisut mahdollistavat eri valmistajien kenttälaitteiden integroinnin samaan järjestelmään, eli asiakas ei ole riippuvainen yhdestä järjestelmätoimittajasta (interoperable = keskinäinen käytettävyys). Integroinnin seuraava taso olisi keskinäinen vaihdettavuus. Keskinäisen vaihdettavuuden ongelmana on standardin puute. Tarvittaisiin standardikieli, jolla laitteet kuvattaisiin yhdenmukaisesti (DDL, Device Definition Language). Laitekuvauskielellä määritellään ohjausjärjestelmän anturien ja toimilaitteiden viestimäärittelyt ja laitetiedot, jotka mahdollistavat kenttälaitteiden kehittämisen standardin mukaisesti [7].

2.4.2 Kenttäväylästandardi

Yhtenäistä kenttäväylästandardia ei ole saatu aikaiseksi, vaan käyttäjillä on tällä hetkellä useita vaihtoehtoja. Ongelmia on ilmentynyt erityisesti eurooppalaisten ja amerikkalaisten näkemyksissä tulevaisuuden kenttäväylästä. Jotkut kenttäväylästandardit ovat kuitenkin muodostumassa teollisuusstandardeiksi. Standardoinnin kehityksen vaikeus johtanee siihen, että markkinoille jää useita kenttäväyläratkaisuja, joista osa on avoimia ja osa toimittajakohtaisia [7].

Euroopassa merkittävintä standardointityötä tekevät CEN (Comité Européen de Normalisation) ja CENELEC (Comité Européen de Normalisation Electrotechnique), joiden alla toimii useita alakomiteoita, jotka tekevät tietyn osa-alueen standardointia [6]. Eurooppalainen kenttäväylästandardi EN 50170 sisältää saksalaisen Profibusin, ranskalaisen FIP:n (Factory Instrumentation Protocol) ja tanskalaisen P-Netin. Euroopan Yhteisön määräysten perusteella tietyn kokoisissa julkisissa urakoissa ei saa käyttää EuroNormiin kuulumattomia toimittajia. Tämä tarkoittaa sitä, että CAN (Controller Area Network) ja muut amerikkalaiset jäävät paitsioon Euroopassa tehtävistä hankkeista. Edellä mainittu standardi on kuitenkin väliaikainen siihen asti, kun varsinainen IEC-standardi (International Electro-technical Commision) valmistuu. CENELEC lähti standardin kehitykseen mukaan, koska IEC-standardin kehitys eteni

hitaasti. IEC-standardista on valmiina ainoastaan OSI-mallin fyysinen kerros. Standardoinnin hidas kehitys on johtanut myös siihen, että laitevalmistajat ovat luomassa yhtenäistä laiteprofiilia, jolloin kenttäväyläratkaisun valitseminen ei ole merkittävässä asemassa, vaan laitteita ohjattaisiin muuten yhtenäisillä tavoilla. Mm. Interbus-S-väylään on tehty jo jonkin aikaa erilaisia profiileja. Kuvassa 7 ovat kenttäväylän eri standardointijärjestöjen ja yritysten väliset yhteydet [3, 8].



Kuva 7. Eri standardointijärjestöjen väliset yhteydet.

Monilla teollisuuden aloilla on oman alan erityisominaisuuksia tukeva väylä. Tämä voi johtaa siihen, että tulevaisuudessa tehtaissa yhdistellään eri valmistajien kenttäväyläsovelluksia, jolloin kenttäväylien yhdistäminen (gateway) on merkittävässä asemassa [3].

Monet käytössä olevat kenttäväyläratkaisut käyttävät tehokkuussyistä johtuen ainoastaan osaa OSI-mallin mukaisista kerroksista. Kenttäväylästandardikin perustuu kolmeen OSI-mallin kerrokseen. Käytössä ovat fyysinen-, siirtoyhteys- ja sovelluskerrokset. Kerrokset 3–6 on jätetty pois reaaliaikavaatimusten vuoksi. Ylimääräisenä kerroksena käytetään käyttäjäkerrosta (User layer), jolla toteutetaan hajautettu ohjaus. Tällä hetkellä standardista on valmiina ainoastaan fyysinen kerros ja siirtoyhteyskerros on juuri valmistumaisillaan. ISA (Instrument Society of American) on jo standardoinut siirtoyhteyskerroksen [23, 24].

Taulukossa 2 on eri kenttäväylätuotteiden käyttökohteita. Tässä työssä käsitellään tarkemmin LON- ja CAN-väyliä. LON-väylään tutustutaan tarkemmin sen vuoksi, että sitä on käytetty verkkokonfiguraattorin pilotissa. CAN-väylä on otettu mukaan vertailun ja väylän suosion vuoksi.

Taulukko 2. Eri kenttäväylien käyttökohteita [24].

Sensorbus	Devicebus	Fieldbus
CAN	CAN	IEC 1158/ISA SP50.02
AS-Interface	DeviceNet	Foundation Fieldbus
InterBus Sensor Loop	SDS	Profibus-PA
Seriplex	CAL/CANopen	Profibus-FM
SERCOS	CAN Kingdom	WorldFIP
Sensoplex	InterBus-S	P-NET
	Device WorldFIP (DWF)	Measurement Bus
	FIP IO	Bitbus
	Profibus-DP	
	II/O Lightbus	
	SERCOS	
	MIL-STD-1553	
	LonWorks	

2.4.3 LON

Local Operating Network (LON) on Echelonin kehittämä hajautettu kenttäväyläratkaisu, jota käytetään teollisuudessa ja erityisesti rakennusautomaatiossa [26]. LONin kehitystyö alkoi vuonna 1986, ja Echelon-yritys on perustettu vuonna 1988. Väylällä käytettävän protokollan nimi on LonTalk Protocol, ja väyläratkaisun nimi on virallisesti LonWorks Network. Tässä työssä käytetään edellä mainituille käsitteille nimiä LON-protokolla ja LON-väylä [28].

LON-väylän jokaisella laitteella tulee olla mikroprosessori, jota kutsutaan neuronpiiriksi. Neuronpiirit ovat varta vasten LON-väylää varten tehtyjä prosessoreita, ja laitteet yksilöidään nimen perusteella (48-bittinen Neuron-ID) [9, 10]. Neuronpiirissä on kolme mikroprosessoria, joilla jokaisella on oma tehtävänsä. Prosessorit ovat mediaohjainprosessori (Media Access Control processor, MAC), sovellusprosessori (Application processor) ja verkkoprosessori (Network processor). Prosessoreita valmistavat tällä hetkellä Motorola ja Toshiba. Eri valmistajien tuotteiden eroja ovat mm. piirien muistimäärä ja jalkajärjestys. Muista kenttäväyläratkaisuista poiketen LON-protokolla käyttää kaikkia OSI-mallin seitsemää kerrosta [28].

Neuronpiirejä ohjelmoidaan Neuron C -ohjelmointikielellä, joka laajentaa ANSI C:tä, jotta kehitettäviin hajautettuihin sovelluksiin saataisiin olio-perustainen lähestyminen. Neuron C tukee suoraan LON-protokollassa käytettäviä verkkomuuttujia ja SNVT:tä.

LON-väylään liittyneiden laitteiden yhdistämiseen voidaan käyttää erilaisia tapoja. LON-protokolla tukee useita langallisia ja langattomia kommunikointitapoja. Perinteisesti käytetään halpaa parikaapelia, mutta mediana voivat toimia myös sähköverkko, radiolinkki, infrapuna-aallot sekä koaksaali- ja valokaapeli. Käytettävästä mediasta riippuen väylän kommunikointinopeus on 5 kbit/s–1,25 Mbit/s. Radioaalloilla ei saavuteta yhtä suuria nopeuksia kuin esim. parikaapelilla.

LON-protokollassa osoitehierarkiassa on kolme tasoa, alue (domain), aliverkko (subnet) ja solmu (node). Alueella voidaan esim. määritellä radioaaltoja käytettäessä tiettyä taajuuskaistaa käyttävät solmut. Yksi solmu voi olla korkeintaan kahdessa alueessa. Yhdessä alueessa voi olla 255 aliverkkoa ja yhdessä aliverkossa voi olla 127 solmua. Näin yhdessä alueessa voi siis olla korkeintaan 32 385 solmua ($255 \cdot 127 = 32\,385$). Solmut voidaan myös jakaa ryhmiin (group) tai osoituksessa voidaan käyttää yksilöllistä tunnistenumeroa. Tätä tunnistenumeroa (Neuron-ID) käytetään tyypillisesti asennuksen ja konfiguroinnin aikana [11].

Verkkomuuttujat LON-väylässä

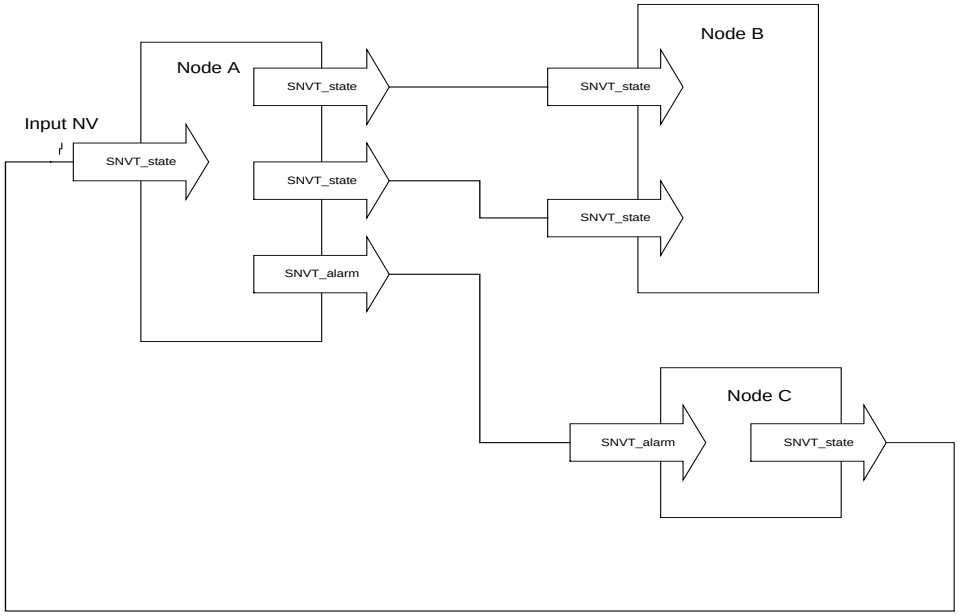
Verkkomuuttujia käytetään tiedon välittämiseen solmujen välillä LON-verkossa. Verkkomuuttujat ovat loogisia muuttujia ja verkon suunnittelijan määrittelemiä. Echelon on määritellyt standardin verkkomuuttujalistan (SNVT MasterList), jossa jokaiselle SNVT:lle on annettu yksiköt, vaihtelurajat ja resoluutio. Listalla on 250 erilaista tyyppiä. Näitä ei ole kuitenkaan pakko käyttää, vaan ne on tehty helpottamaan suunnittelijan työtä, koska muuttujat ovat hyvin usein juuri SNVT-muuttujien tyyppisiä [11]. Taulukossa 3 on esimerkki SNVT MasterList -listasta [12]. Laitteen konfigurointia varten on kehitetty oma lista (SCPT MasterList). SCPT ei kuluta verkkomuuttujille varattua tiedonsiirtoresursseja, vaan se ladataan solmuun tai solmusta aina tiedoston siirto -protokollalla (file transfer). Tyyppejä on 30 erilaista [13].

Taulukko 3. Eräs standardisoitu verkkomuuttujatyyppi.

Mittauskohde	Nimi	Vaihteluraja (Resoluutio)	SNVT #
Taajuus	SNVT_freq_hz	0–6553,5 Hz (0,1 Hz)	76

LON-väylässä voidaan lähettää ja vastaanottaa kahdenlaisia viestejä, eksplisiittisiä ja verkkomuuttujaviestejä. Verkkomuuttujaviestit ovat rajallisen kokoisia (korkeintaan 31 tavua), kun taas eksplisiittiset viestit voivat olla suurempiakin. LON-väylä ei prosessoi eksplisiittisiä viestejä vaan välittää sen ylemmille kerroksille kohdesolmussa. Eksplisiittiset viestit voivat sisältää verkon hallinta- ja konfigurointitietoja, joilla Neuron Chipiä voidaan konfiguroida.

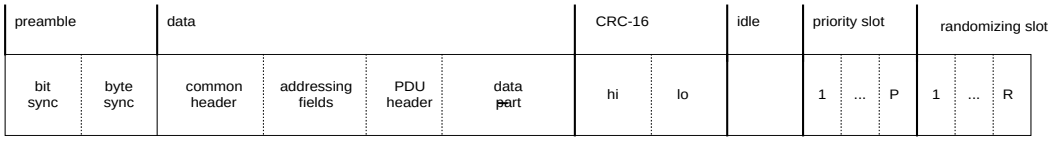
Kuvassa 8 [9] on esimerkki verkkomuuttujien käytöstä. Tiedonkulku solmusta toiseen on hoidettu verkkomuuttujien avulla. Verkkomuuttujia kuvassa 8 ovat SNVT_state ja SNVT_alarm. Solmusta toiseen ei kulje tietoa jatkuvasti, vaan tietoja päivitetään viestien avulla verkon suunnittelijan määräämällä tavalla. Luvussa 5 perehdytään verkkomuuttujien sitomiseen (binding) ja selvitetään tarkemmin sitomisessa tarvittavia attribuutteja.



Kuva 8. Verkkomuuttujien käyttö LON-verkossa.

LON-protokollan viestin rakenne

Kuvassa 9 on yleinen LON-protokollan viestin rakenne. Viestit lähetetään käyttämällä differentiaalista Manchester-koodausta. Viestin alussa on Preamble-osuus, joka on synkronointia varten. Sen jälkeen lähetetään Data, joka sisältää otsikon, osoitteen ja varsinaisen viestin. Virheenkorjauksessa käytetään CRC:tä (Cyclic Redundancy Check). Viestille on mahdollisuus valita myös prioriteetti ja, jos lähetetään prioriteetitön viesti, niin Randomizing Slot määrää, milloin tällaisen viestin lähetys aloitetaan [9].



Kuva 9. Viestin rakenne LON-väylässä.

2.4.4 CAN

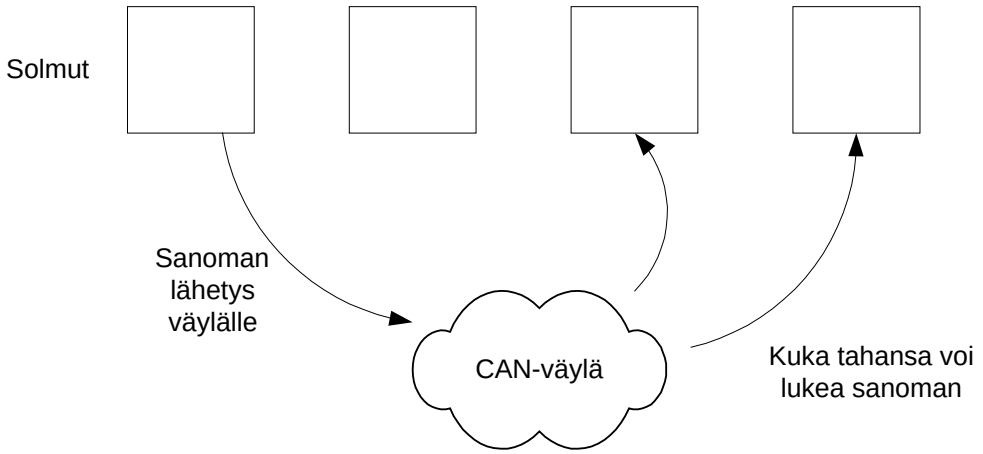
CAN-väylää (Controller Area Network) sovelletaan teollisuudessa erityisesti erilaisissa laitteissa ja koneissa, ei niinkään tehdasväylänä. CAN-väylän viestijärjestelmä on hyvin samantapainen kuin LON-väylässä eli viestit lähetetään yleislähetteenä ja ainoastaan kohdesolmu tai solmut kopioivat sen itselleen.

Korkeamman tason CAN-protokollat ovat CAL, CAN Kingdom, CANOpen, DeviceNet ja Smart Distributed System (SDS). SDS on tarkoitettu ensijaisesti käytettäväksi ohjaus järjestelmissä ja I/O-laitteiden välisessä isäntä-renki-tyyppisessä tiedonsiirrossa. Allen-Bradleyn DeviceNet perustuu CAN-teknologiaan ja sitä käytetään anturi- ja toimilaitetason viestinvälityksessä. Varsinainen CAN on tarkoitettu lähinnä alemman tason tiedonsiirtoon. CAN on kulkuneuvoissa käytettävistä kenttäväylistä suosituin. Väylää käytetään mm. junissa, laivoissa ja lentokoneissa.

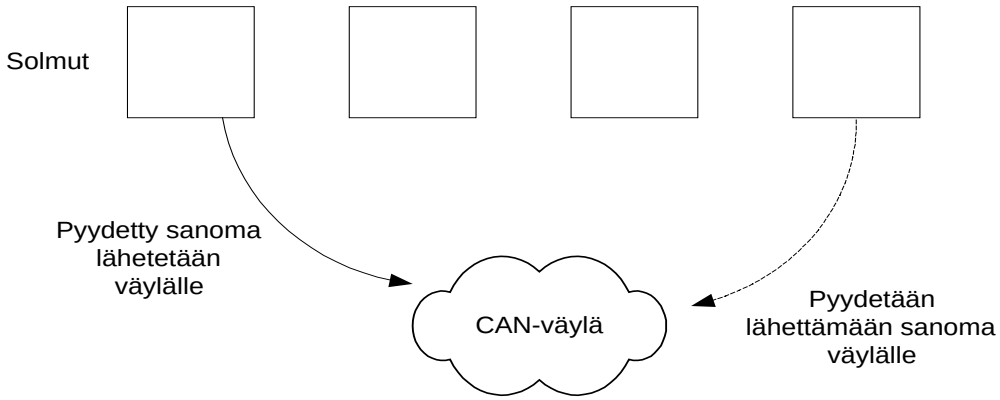
CAN-väylän tiedonsiirtonopeus on valittavissa väliltä 125 kb/s–1 Mb/s. Siirtoetäisyys nopeudella 1 Mb/s on maksimissaan 40 metriä ja nopeudella 125 kb/s 500 metriä [8].

CAN-protokolla tarjoaa peruspalveluna 8 tavun viestien lähetyksen ja vastaanottamisen. Käytännössä peruspalveluiden lisäksi tarvitaan myös ylemmän kerroksen palveluita, jotka joudutaan toteuttamaan ohjelmistoilla. Mainittuja ylemmän kerroksen palveluja ovat esim. lohkosiirto, kuittauksellinen tiedonsiirto ja solmuvalvonta. ISO on standardoinut väylän fyysisen ja siirtoyhteyskerroksen.

Lähetettävässä sanomassa ei ole lähettäjän tai vastaanottajan osoitetta vaan sanoman numero. Kuvassa 10.a on sanoman lähetyksen ja vastaanoton periaate. Jokin anturi voi lähettää esim. jänniteviestiä, jonka numero on 300. Kaikki ne solmut, jotka tarvitsevat jänniteviestiä, kopioivat kyseisen viestin itselleen. Samalla tunnisteella lähettäminen on kiellettyä, koska silloin viestit törmäävät väylällä ja tätä CAN-protokolla ei salli. CAN-väylässä on mahdollisuus käyttää ns. kyselykehystä, jolla voidaan pyytää jokin solmu lähettämään haluttu sanoma väylälle. Protokolla sallii saman kyselykehyksen lähettämisen yhtä aikaa. Kyselyn periaate kuvassa 10.b. CAN-väylässä olevien asemien lukumäärä on tyypillisesti maksimissaan 16, mutta tietyillä muutoksilla lukumäärä voi olla 100 [21, 29].



a) Sanoman lähetys ja vastaanotto.



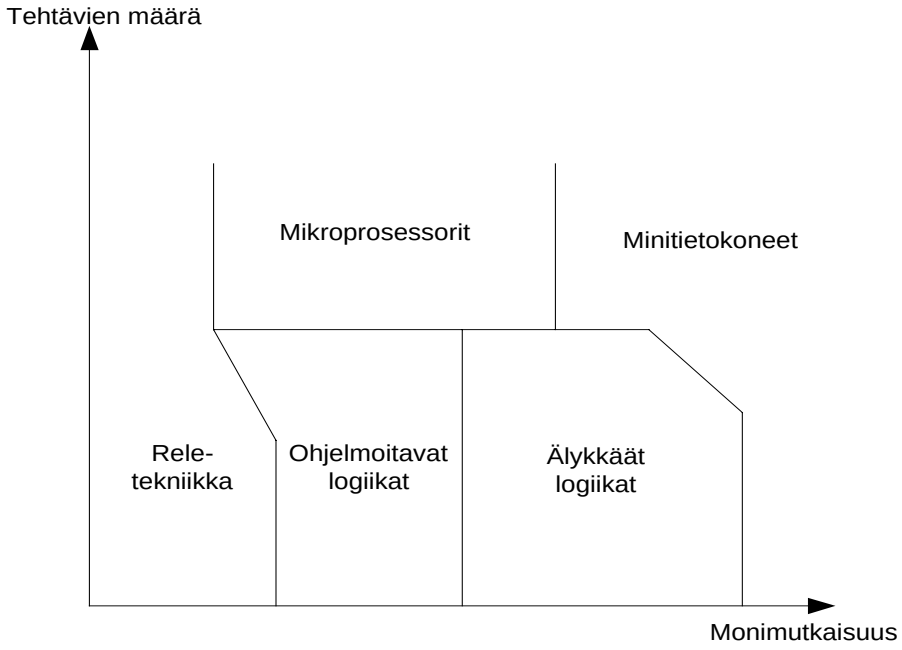
b) Kyselykehyksen lähettäminen.

Kuva 10. CAN-väylän sanoman lähetys.

3. SOVELLUSOHJELMOINTI JA KONFIGUROINTITARPEET

3.1 Laiteympäristöt

Automaation ohjausjärjestelmien laskenta ja logiikka toteutetaan hyvin paljon käyttäen mikroprosessoripohjaisia laitteita. Mikroprosessoripohjaiset laitteet ovat korvanneet perinteisen relelogiikan miltei kokonaan. Yleisimmin ohjausjärjestelmissä käytetään ohjelmoitavaa logiikkaa (PLC, Programmable Logic Controller), vaikka käytössä on jopa mikro- ja minitietokoneita. Rele- ja mikroprosessoreihin perustuvien ohjausjärjestelmien soveltuvuus tehtävien määrän ja monimutkaisuuden funktiona on esitetty kuvassa 11.



Kuva 11. Ohjausjärjestelmissä käytettävät laitetekniikat.

Logiikan toiminta on seuraava: loogiset toiminnot, jotka halutaan ohjauksessa toteutettavan, sijaitsevat ohjelmamuistissa. Keskusyksikkö suorittaa näitä loogisia käskyjä, joiden operaattoreina piirin tulot toimivat. Tuloksena saadaan piirin lähdöt. Kaikki säännöt ovat siis muistissa, eikä mitään fyysisiä portteja näin tarvita.

Ohjelmoitava logiikka sisältää vähintään seuraavat yksiköt:

1. tuloyksiköt
2. lähtöyksiköt
3. keskusyksikkö
4. ohjelmamuisti ja
5. jännitteen syöttö.

Ohjelmoitavien logiikoiden mikroprosessorit (keskusyksikkö) on valmiiksi ohjelmoitu niin, että ne ymmärtävät yksinkertaista logiikkakieltä. Vaikka laitteet sisältävät elektronisia piirejä, niiden käyttö ei vaadi kuitenkaan elektroniikan tuntemusta. Ohjelmat kirjoitetaan ohjelmamuistiin ohjelmointilaitteen avulla. Samaa laitetta voi käyttää testaukseen ja vikojen paikallistamiseen [27].

PLC-laitteiden kommunikoinnissa ulkomaailmaan käytetään erittäin paljon RS-232-sarjaliikenneporttia [30]. Jos on tarvetta hajautettuun järjestelmään, paras vaihtoehto on kenttäväylän käyttö kommunikaation perustana [15]. Nykyään voidaan käyttää myös LAN-verkkoa (Ethernet ja TCP/IP) ohjausjärjestelmien hajautusratkaisuna, vaikka se teknisesti ei yleensä olekaan optimaalinen, mutta se on kaupallisesti tuettu ja stabiili ratkaisu.

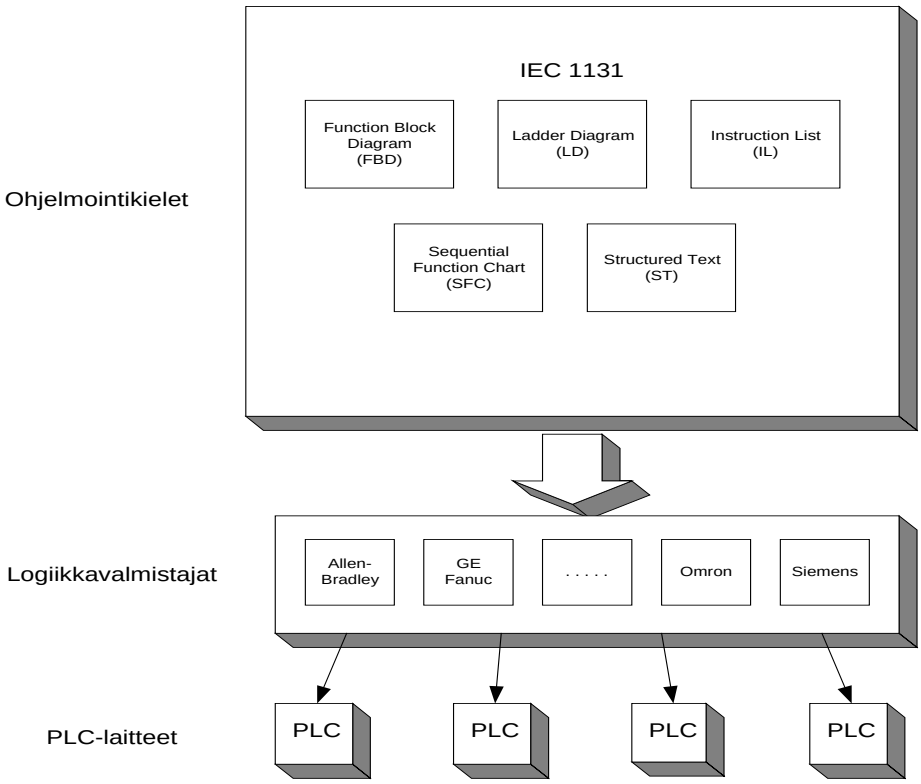
3.2 Ohjausjärjestelmäsovellusten ohjelmointi

3.2.1 Taustaa

Standardin tarve on ollut ilmeinen ohjainsovellusten ohjelmoinnissa, koska sovellusten kehittämisessä on ollut käytössä niin monia eri ohjelmointikieliä. Ohjelmoinnissa on käytetty viimeisten 10–15 vuoden aikana mm. BASIC-, FORTH-, C-, structured English ja Instruction List -kieliä. Standardin kehitystyön tuloksena on syntynyt logiikan ohjelmointityökalujen standardi, IEC 1131-3. Kehitystyö standardin luomiseksi aloitettiin IEC:n toimesta vuonna 1979. Työryhmän tavoitteena oli määrittää PLC:n ohjelmointiproseduuri, joka sisältäisi kovon suunnittelun, asennuksen, testauksen, dokumentoinnin, ohjelmoinnin ja tiedonsiirron [15].

3.2.2 Standardi IEC-1311

Standardissa määritellään 5 eri kieltä, jotka ovat perustana PLC-ohjelmointi-ympäristöille. Kuvassa 12 on standardin toimintaperiaate.



Kuva 12. Standardin IEC 1131-3 toimintaperiaate.

SFC (Sequential Function Chart) on graafinen kuvauskieli, jolla voidaan kuvata kaaviona ohjelman sisältämät sekvenssit. Askeleet (stepit) ja siirtymät ovat SFC:n perustoimintoja. Muilla IEC 1131-3 -standardin kielillä kuvataan askeleiden ja siirtymien sisältö. IL (Instruction List) on matalan tason ohjelmointikieli, jota voidaan käyttää pienissä sovelluksissa tai jonkin ohjelman osan optimointiin. Structured Text (ST) on ohjelmointikieli, jolla voidaan ilmaista funktioiden toimintaa, funktiolohkoja ja ohjelmia. Kieli on korkean tason kieli ja muistuttaa hyvin paljon PASCALia. Vaikka ST muistuttaa PASCALia, se on kuitenkin tehty nimenomaan ohjainsovelluksia varten. Kielen käskyjen kirjoittaminen on melko vapaata ja tämä osaltaan vaikuttaa siihen, että kieltä on helppo opiskella ja käyttää. ST on erittäin käytännöllinen, kun tarvitaan kompleksisia aritmeettisia laskutoimituksia. Tikapuu- tai relekuvaus eli LD (Ladder Diagram) käyttää standardoitua relekaaviosymbolikokoelmaa. Function Block Diagram (FBD) -ohjelmointikielellä ilmaistaan funktioiden toimintaa, funktioita ja ohjelmia lohkojen väliin piirrettyjen yhteyksien avulla. Kieli on siis graafinen ja muistuttaa signaalivirtojen piirtämistä sähköpiiridiagrammeissa [14, 15].

3.2.3 PLC-sovellusten ohjelmointityökaluja

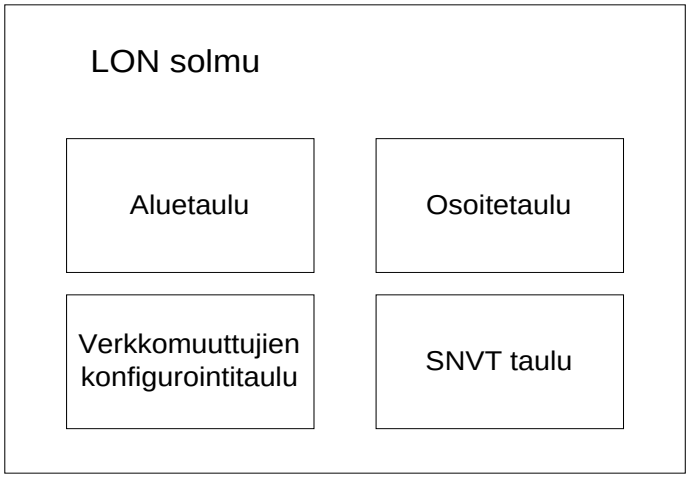
Kaikki PLC-sovellusten ohjelmointityökalut perustuvat IEC 1131 -standardiin muutamaa poikkeusta lukuun ottamatta. Standardi IEC 1131 on tehnyt mahdolliseksi yleisten PLC-työkaluohjelmistojen tuotannon, joista esimerkkinä voidaan mainita CJ Internationalin ISaGRAPH ja Wizdom Controls Inc:n Paradym-31. Nämä ohjelmistot pyrkivät noudattamaan mahdollisimman tarkasti standardia, kun taas jotkut työkalut noudattavat osittain IEC 1131 -standardia. Osittain standardia noudattavat työkalut on suunniteltu käytettäväksi milteipä yksinomaan työkalun kehittämän yrityksen logiikoiden ohjelmoimiseen [14].

3.3 Hajautettujen ohjainsovellusten konfigurointi

3.3.1 Konfigurointitarpeet

Konfiguroinnilla tarkoitetaan toimintojen lisäämistä ja poistamista, kommunikaation ja I/O-tietojen määrittelemistä verkkoon liittyneille laitteille. Konfiguroinnin avulla verkko saadaan toimimaan halutulla tavalla.

Esimerkiksi LON-verkon konfiguroinnissa käyttäjä määrittelee ensiksi solmujen sisäiset parametrit ja tämän jälkeen manuaalisesti jokaisen solmun liitynnän. Liityntöjen konfigurointi on käytännössä solmun osoitteiden ja verkkomuuttujien muodostamista, joka tarkoittaa neuronpiirissä olevien osoite (address table) ja verkkomuuttujien konfigurointitaulujen (Network Variable Configuration Table, NVCT) täyttämistä [19]. Kuvassa 13 [9] on neuronin muistin sisältämät taulut.



Kuva 13. Neuronpiirin sisältämät taulut.

3.3.2 Dynaaminen konfigurointi

Järjestelmä voidaan konfiguroida sen ollessa kytkettynä tai poiskytkettynä. Pyrkimys on tehdä kaikki konfigurointi kytkettynä kustannusten minimoimiseksi ja, koska järjestelmän uudelleenkäynnistäminen on aina vikaherkkää.

Hajautettujen sovellusten konfigurointi käsittää seuraavat operaatiot:

1. alkukonfiguroinnin suorittaminen.

2. hajautetun sovelluksen toiminnanylläpito

3. yhdenmukaisen tilan tarjoaminen sovellukselle koko toiminnan ajan.

Konfiguraation muutokset sisältävät komponenttien siirtämistä ja poistamista sekä solmujen välisien yhteyksien muuttamista. Näiden operaatioiden suorittaminen saisi häiritä sovelluksen toimintaa mahdollisimman vähän. Tämä onkin hajautettujen sovellusten konfiguroinnin hallinnan päätavoite. Konfigurointi tulisi suorittaa mahdollisimman läpinäkyvästi sovelluksen kannalta [25].

Hyvältä konfiguroinnin hallinnalta vaaditaan ainakin seuraavia ominaisuuksia:

Muutosten määrittelyjen tulisi olla korkeatasoisia.

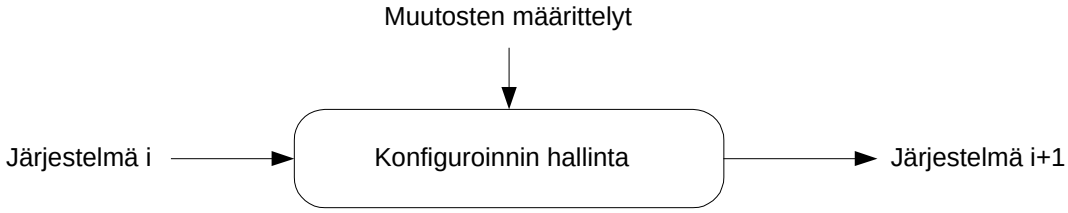
1. Muutosten määrittelyjen tulisi olla selittäviä.

2. Muutoksien tulisi olla riippumattomia järjestelmässä toimivista sovelluksista.

3. Järjestelmän tulee pysyä yhdenmukaisessa tilassa.

4. Suorituskykyyn ei saa tulla huomattavia muutoksia.

Konfiguroinnin yhteydessä erityisesti yhdenmukaisuuden säilyttämisen varmistaminen on hankalaa. Jotta järjestelmä olisi yhdenmukainen, sen tulisi säilyttää tietyt arvot tosina koko konfiguroinnin ajan. Kuvassa 14 on järjestelmän ideaalinen konfigurointi, eli järjestelmä toimii moitteettomasti, kun siihen lisätään komponentteja. Yhdenmukaisuuden säilyttämiseksi järjestelmässä on ainakin seuraavat kolme eri tapaa: epäyhdenmukaisuuden välttäminen (avoidance) keskeyttämisen avulla, epäyhdenmukaisuuden välttäminen tilan säilyttämisellä ja epäyhdenmukaisuuden korjaaminen.



Kuva 14. Järjestelmän ideaalinen konfigurointi.

Avoidance through suspension -menetelmä perustuu siihen, että muutokset tulisi tehdä staattisessa ja yhdenmukaisessa tilassa sen jälkeen, kun osa käynnissä olevista prosesseista on keskeytetty. Menetelmän proseduuri yleisellä tasolla on seuraavanlainen:

1. Konfiguroinnin hallintaprosessi (Configuration Management Process, CMP) lähettää signaalit kaikille konfiguraatiomuutoksiin liittyville prosesseille, jotta ne keskeyttäisivät toimintansa. Keskeytetyt prosessit siirtyvät aktiivisesta tilasta ns. nukkuvaan tilaan odottamaan uudelleenaktivointia.

2. CMP suorittaa muutokset järjestelmään.

3. CMP aktivoi keskeytetyt prosessit.

Vaiheessa 3 aktivoitujen prosessien ei tulisi luoda uusia yhteyksiä prosessiin, johon tehtiin muutoksia kohdassa 2. Samalla prosessien tulisi säilyä yhdenmukaisena. Edellä mainittu vaatimus rajoittaa melko paljon sovelluksia, jotka pystyvät käyttämään Avoidance through suspension -menetelmää. Konfigurointiproseduurin tehokkuus riippuu paljolti keskeytettyjen prosessien lukumäärästä vaiheessa 1.

Toinen tapa välttää epäyhdenmukaisuus on kopioida prosessin vanha versio sillä aikaa, kun se on konfiguroitavassa tilassa. Jotta prosessi saavuttaisi konfiguroitavan tilan, sen kommunikointikanavat tulee olla keskeytettynä. Tällä menetelmällä ei saavuteta yhtä hyvää suorituskykyä kuin Avoidance through suspension -menetelmällä. Menetelmän aiheuttama alhaisempi suorituskyky saattaa aiheuttaa uusia ongelmia [16].

3.3.3 LON-verkon konfigurointi LNT-työkalulla

LON-verkossa konfiguroinnilla voidaan määritellä uusia solmuja verkkoon tai poistaa tai muuttaa olemassa olevia asetuksia. Lisäksi voidaan konfiguroida sovellusten välillä kulkevia tietoja ja rajapintoja eli käytännössä verkkomuuttujia ja niihin liittyviä attribuutteja.

Laitteiden installointia ja konfigurointia LON-verkossa voidaan tehdä esimerkiksi LNT-työkalulla (Lon Network Tool), joka mahdollistaa LON-verkon manuaalisen konfiguroinnin. LNT-työkalu on suomalainen ohjelmisto, jonka on kehittänyt ja toteuttanut ABB Transmit Oy yhteistyössä SWE Oy:n (Software Engineers Oy) kanssa. LNT-työkalua käytetään LON-pohjaisten relelaitteiden integroimiseksi toisiinsa.

LNT-työkalulla luodut projektit (konfiguroinnit) talletetaan tietokantaan. Nämä konfigurointitiedot voidaan aina tarvittaessa ladata muokattavaksi työkaluun tietokannasta. Työkalulla konfigurointi voidaan tehdä joko kytkettynä tai kytkemättömänä.

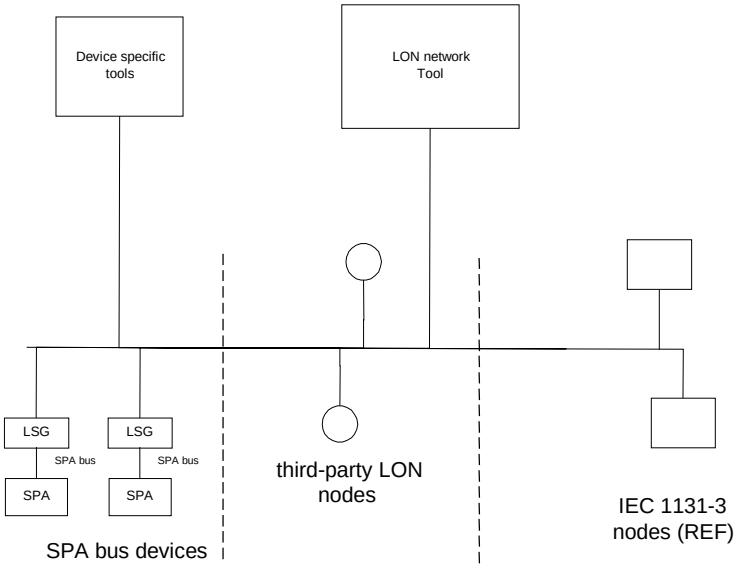
Työkalulla voidaan tarkkailla LON-verkon laitteiden tiloja tiettynä hetkenä eli sitä voidaan käyttää myös verkon valvontaan. Työkalu voidaan liittää johonkin tuntemattomaan verkkoon, jos halutaan saada selville sen rakenne. Toimintojen suorittamiseen käyttäjällä on usein mahdollisuus valita monta eri tapaa suoriutua tehtävästä. Esimerkiksi Domain Id -attribuutin muuttamiseksi voidaan käyttää ainakin kahta eri tapaa [17].

Konfigurointitietojen siirtäminen tietokannasta LON-väylän fyysisiin laitteisiin voidaan tehdä käyttämällä hyväksi esim. NETAgent-ohjelman palveluja. NETAgent on saman yrityksen tuote kuin LNT. NetAgent käyttää DDE-liityntäkuvausta (Dynamic Data Exchange). DDE on kahden sovelluksen välinen kommunikointitapa, joka käyttää hyväkseen jaettua muistia (shared memory). DDE-liitynnän käyttö vaatii keskustelujen muodostamista asiakkaan (tässä tapauksessa LNT:n) ja palvelimen (NETAgentin) välille [18].

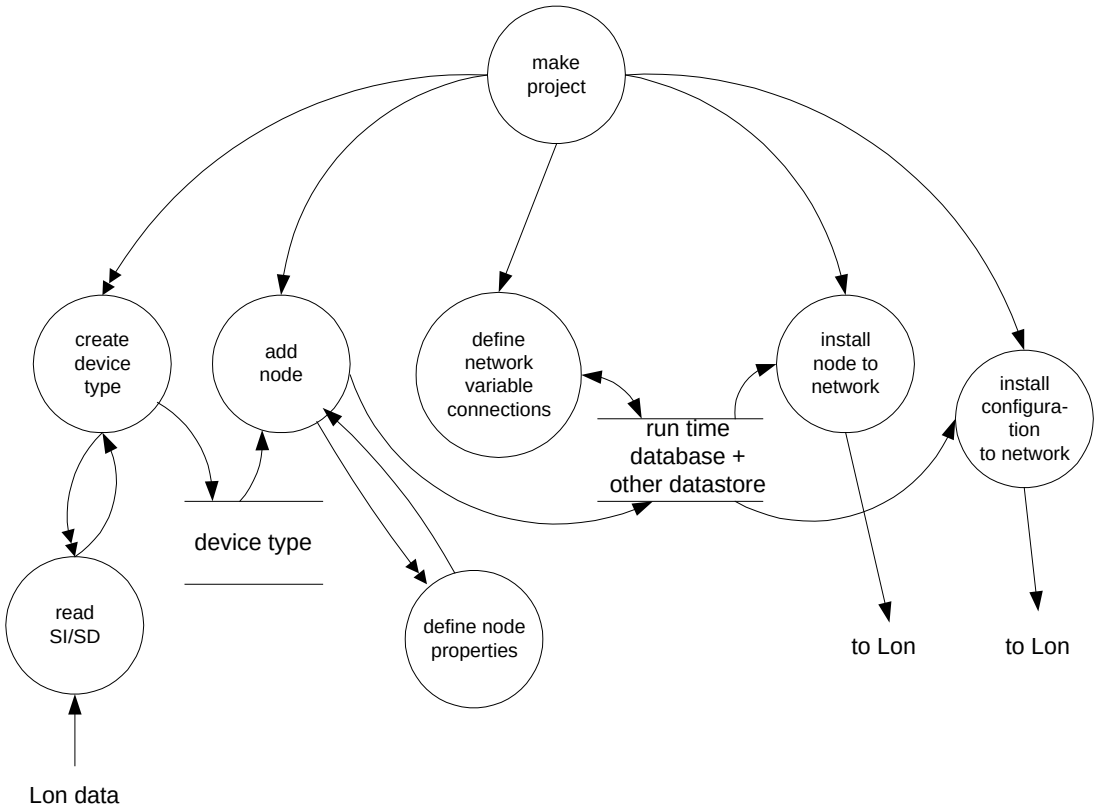
Kuvassa 15 on erityyppisiä solmuja ja niiden konfigurointiin tarkoitettuja työkaluja. LON-verkon laitteen installointi- ja konfigurointiproseduuri on karkeasti seuraava:

1. laitteen asennus verkkoon
2. peruskommunikaation määrääminen
3. solmun sisäinen ohjelmointi, konfigurointi ja parametrisointi
4. solmun liityntöjen konfigurointi

Kuvassa 16 on SA/SD-kaavio konfigurointiproseduurissa. Kaaviossa on määritelty tarkemmin konfigurointiproseduurissa tarvittavat toiminnot. Suurin ongelma proseduurissa on verkkomuuttujayhteyksien määritteleminen (define network variable connections) [19]. Seuraavassa kohdassa 3.4 tarkastellaan tarkemmin konfigurointiin liittyviä ongelmia.



Kuva 15. Eräs mahdollinen LON-väyläpohjainen järjestelmä.

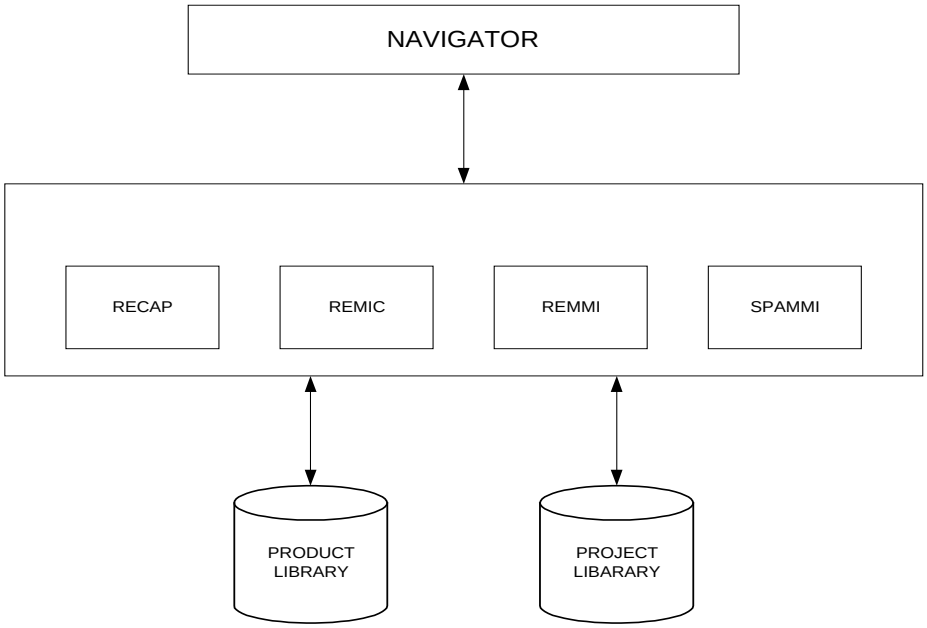


Kuva 16. LNT:llä suoritettavat funktiot konfiguroinnissa

3.4 Konfiguroinnin automatisoinnin kehittämistarpeet

Konfiguroinnin suorittaminen LNT-työkalulla on työlästä ja virheherkkää, koska konfigurointi tehdään manuaalisesti. Tämä pitää paikkansa varsinkin verkon liityntöjen osalta, sillä liityntöjä voi olla erittäin paljon, jopa tuhansia [19]. Yleisesti ottaen konfiguroinnin automatisoinnin vaikeutena on konfigurointien standardoinnin puute.

Lisäksi ongelmia aiheuttaa työkalujen moninaisuus. Verkon toteuttamiseen alusta loppuun tarvitaan monia erilaisia suunnittelu- ja konfigurointityökaluja. Esim. ABB Transmit Oy:llä käytetään releiden ja releitä käyttävien järjestelmien suunnittelussa CAP 505 -työkalupakettia (Computer Aided Programming), johon kuuluu 4 eri työkalua. Kuvassa 17 on CAP 505 -tuoteperheeseen kuuluvat työkalut.



Kuva 17. CAP505-työkalupaketti.

Kuvassa 17 Product Library sisältää laitetyyppien kuvaukset, kun taas Project Library sisältää projektikohtaiset järjestelmäsuunnittelun kuvaukset. RECAP-työkalu (RElay Configuration And Programming) on laitteen IEC 1131 mukainen ohjelmointityökalu, joka tukee suunnittelua projektitasolla eli integrointitiedon luontia. Navigatoria käytetään projektinhallintaan.

Koska sovellukset sisältävät hyvin paljon solmuja ja niiden välisiä liityntöjä, myös työkalujen kapasiteetti rajoittaa suunnittelua. Esim. visuaalinen sovelluksien suunnittelu

vetämällä yhteydet solmujen välille on melko vaikeata, koska kuvaruudulle mahtuu ainoastaan rajallinen määrä solmuja. Suunnittelu tapahtuu hyvin pitkälle siten, että kuvaruudussa on vain yksi laite, jolle nimetään tulot ja lähdöt.

Tarve kehittää työkaluja on ilmeinen. Samoin suunnitteluproseduuria voi systematisoida ja suunnittelutyökaluja yhtenäistää.

Ideaalisella työkalusetillä pitäisi pystyä

- suunnittelemaan automaatiolaitteet ja niiden väliset liitynnät
- asentamaan laitteet ja konfiguroimaan ne ajonaikaisesti
- valvomaan verkon laitteiden toimintaa.

4. KONFIGUROINNIN AUTOMATISOINTI

4.1 Edellytykset

Tässä työssä konfiguroinnin automatisointi liittyy integrointityövaiheen kytkennällisyyskuvausten automaattiseen toteuttamiseen. Automatisoinnilla pyritään vähentämään konfiguroinnissa tapahtuvia virheitä ja nopeuttamaan suoritusta.

Konfiguroinnin automatisointi vaatii keskitetyn tiedohallintajärjestelmän. Tällainen voi olla esim. tietokanta. Kun käytetään keskitettyä tiedonhallintaa, järjestelmä mahdollistaa seuraavat toiminnot:

1. Koska konfigurointi tarvitsee eri vaiheessa tuotettua suunnittelutietoa, on tiedot järkevintä koota yhteen paikkaan. Esim. laite- ja projektitiedot tuotetaan eri työkaluissa eri aikaan.

2. Tietokannan avulla voidaan helposti integroida eri työkalut yhteen. Tällöin ei tarvitse miettiä ratkaisua eri työkalujen välillä vaan ainoastaan liityntä työkalusta tietokantaan.

3. Liityntä tietokantaan on melko helppo toteuttaa, koska markkinoilla on useita toimivia ratkaisuja valmiina.

4. Suunnittelutieto on tietokannassa määrämuodossa, jonka kaikki työkalut tiedostavat.

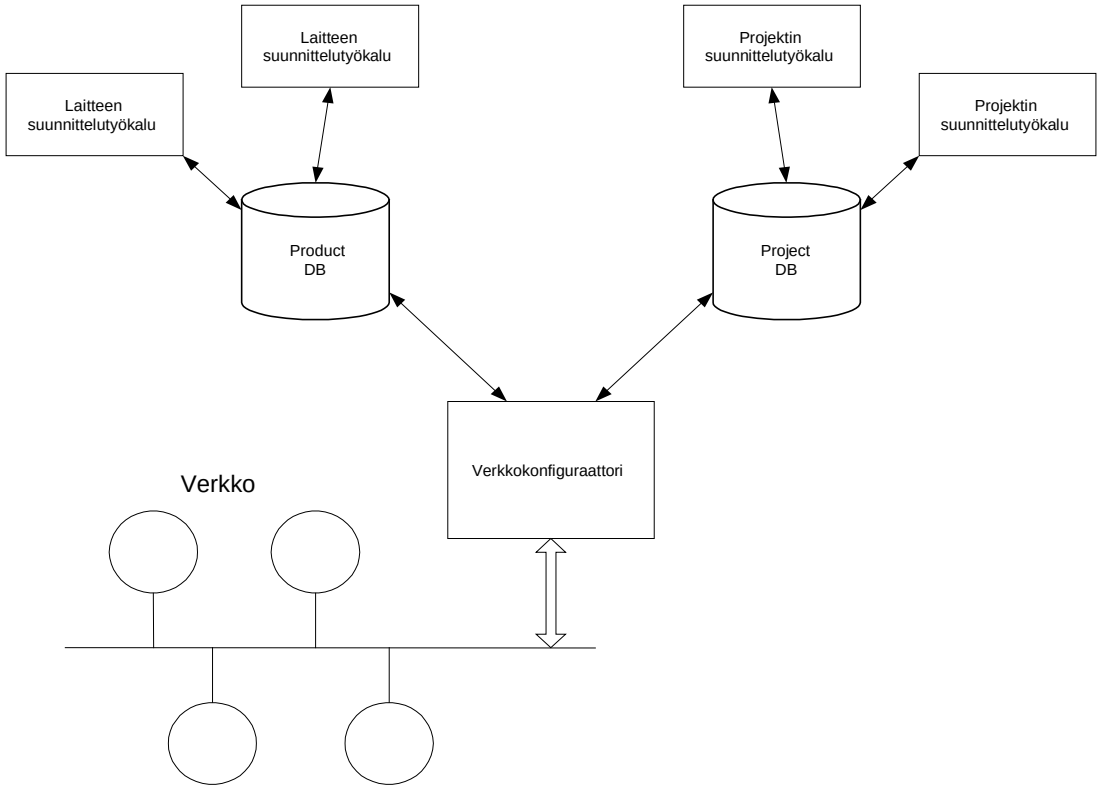
Tärkein edellytys automaattisen konfiguroinnin mahdollistamiseksi on eri työkalujen integroiminen yhteen. Mikäli työkalut ovat jo samaa ”perhettä”, integrointi on helpompaa. Keskitetty tiedonhallintajärjestelmä on edellytys sille, että työvaiheet voitaisiin automatisoida. Ehdottomia edellytyksiä nämä eivät ole, mutta ne helpottavat toteuttamista erittäin paljon.

4.2 Konfiguroinnin automatisointi

Konfiguroinnin automatisointiin liittyviin ongelmiin voi olla ratkaisuna konfigurointityökalujen uudistaminen. Liityntäkuvaukset ja muut verkon konfigurointitiedot tulee tallettaa tietokantaan, josta ne ovat eri työkalujen saatavilla. Laitteiden välisien liityntöjen suunnittelu- ja konfigurointiproseduurissa tapahtuvia redundantteja toimituksia voi vähentää lisäämällä älykkyyttä konfigurointityökaluun. Verkkomuuttujayhteydet voidaan lukea suoraan kannasta sen sijaan, että käyttäjä

syöttäisi ne manuaalisesti uudestaan jossain toisessa työkalussa. Tällä hetkellä verkkomuuttujien määrittely joudutaan tekemään useaan kertaan.

Kuvassa 18 on periaate ratkaisusta. Ratkaisun ydin on verkkokonfiguraattorityökalu, joka suorittaa automaattista konfigurointia verkkosovellukseen. Työkalu sisältää tietokannan, missä yhdistetään laite- ja projektisuunnittelussa tuotetut tiedot (tiedot ovat tällä hetkellä eri tietokannoissa). Päättelysääntöjä hyväksi käyttäen verkon konfigurointi tapahtuu automaattisesti.



Kuva 18. Ratkaisu työkalujen integroimiseksi.

4.3 Keskitetty automaattinen konfigurointi

Keskitetyllä konfiguroinnilla tarkoitetaan tässä yhteydessä konfigurointitietojen keskittämistä yhteen paikkaan. Tietojen keskittäminen on helpointa tehdä tietokannan avulla. Tietokantoja on markkinoilla erittäin paljon useisiin eri käyttötarkoituksiin.

Konfigurointitietojen keskittämisestä on sekä etuja että haittoja. Keskittämisen etuna on se, että tieto on ajan tasalla, eli kenenkään ei pitäisi saada väärää tietoa. Edellytyksenä on kuitenkin se, että päivitykset tehdään ajallaan ja tietokantajärjestelmässä on ratkaistu kriittisen alueen ongelma. Konfiguroinnin automatisoinnin helpottamiseksi tietojen keskittäminen on tarpeen, sillä tietojen saanmisen proseduurin eri vaiheessa tulee olla mutkatonta.

Tällä hetkellä LNT-konfigurointityökalussa käytetään rinnakkain sekä tietokantaa että tiedostoformaattia konfigurointitietojen hallitsemisessa. Tiedostojen muuttaminen tietokannan ymmärtämään muotoon voi olla ongelma, joka ratkeaa, kun käytetään pelkästään tietokantajärjestelmää. Muunnokset formaatista toiseen poistuvat ja ylläpito helpottuu huomattavasti.

Tietokanta toimii palvelimena sitä käyttäville työkaluille (asiakkaat). Tietokantaan voi liittyä monella tavalla. Yksi suosittu tapa liittyä, varsinkin jos käyttää Microsoftin tuotteita, on ODBC-liityntä (Open Database Connectivity). ODBC-tietokantaliitynnästä kerrotaan enemmän kohdassa 5.4.1.

Embedded SQL (ESQL) ja CLI (Call Level Interface) ovat kaksi erilaista tapaa olla yhteydessä tietokantaan. ODBC 1.0 oli ensimmäinen CLI-spesifikaatio. Nykyinen ODBC on kuitenkin laajennettu CLI, sillä ODBC on kolmitasoinen ja CLI vastaa ainoastaan core-tasoa. ESQL:ssä täytyy koodi ajaa esikääntäjän läpi, jotta varsinainen kääntäjä ymmärtää SQL-koodia. CLI ei vaadi esikääntämistä, joten se on joustavampi, mutta hitaampi. ESQL on tehokkain tietokantaliityntä, mikäli käyttää vain yhtä tiettyä tietokantasovellusta.

Stored Procedures -menetelmä vähentää SQL-käskyjen esiprosessointia, ja yksi tietokantatoimitus voi olla jopa viisi kertaa nopeampi kuin ESQL:ssä. Stored Procedures -menetelmää ei ole kuitenkaan standardoitu, vaan jokaisella ohjelmistotoimittajalla on oma tapansa hoitaa SQL-käskyt [35].

ODBC-liityntä on joustava, koska se ei ole riippuvainen tietokannasta. Tällä saavutetaan se etu, että tietokanta on vaihdettavissa helposti johonkin toiseen. Haittana on kuitenkin ODBC:n tulevaisuus, sillä Microsoft ei ole kovin kiinnostunut jatkamaan kehitystyötä. ODBC on kuitenkin erittäin suosittu, ja sen sovelluksia on lukematon määrä.

Relaatiotietokanta soveltuu konfiguraatiotiedon ylläpitoon melko hyvin. Relaatio-tietokantoja on käytetty jo pitemmän aikaa, joten kannat ovat kehittyneitä ja suoritus-kykyisiä.

Monen tiedontalletusmuodon järjestelmästä siirtyminen keskitettyyn järjestelmään saattaa aiheuttaa työkaluissa isojakin muutoksia. Työmäärä työkalujen muuttamiseksi

yhtenäisiksi saattaa olla kallis ja aikaa vievä projekti. Silloin kannattaa miettiä, saavutetaanko automatisoinnilla niin paljon etua, että suuret muutokset nykyisissä työkaluissa kannattavat. Tällaisten asioiden arvioiminen on kuitenkin vaikea tehtävä.

4.4 Itsekonfiguroituvuus

Nykyisin useat laitteet sisältävät erittäin paljon älykkyyttä eli niissä on kehittyneet prosessorit ja muistia. Tämä mahdollistaa tehokkaamman älykkyyden hajauttamisen verkon laitteille kuin perinteisillä I/O-järjestelmillä. Itsekonfiguroituvuudella tarkoitetaan sitä, että laite pystyy liittymään verkkoon itsenäisesti eli laite- ja liityntäkuvaukset olisivat valmiina laitteen muistissa. Verkossa tulee tällöin olla koordinaattori, jolla on ylin päätäntävalta. Ongelmien välttämiseksi koordinaattori ja laite neuvottelevat verkkoon liittymisestä. Neuvottelun perusteella koordinaattori myöntää laitteelle luvan liittyä verkkoon tai kieltää liittymisen. Itsekonfiguroituvuus hyödyntää agenttipohjaista ohjelmistoarkkitehtuuria.

Sovelluksia itsekonfiguroituvuuden käytöstä hajautetuissa ohjausjärjestelmissä ei ole, mutta esim. PC-tietokoneissa käytetään plug and play -tekniikkaa, kun halutaan liittää uusi laitekomponentti järjestelmään. Laite voidaan liittää tietokoneeseen sen ollessa päälle kytkettynä. Tietokone osaa tunnistaa laitteen, ja käyttöjärjestelmä kyselee käyttäjältä tarvittavia asetuksia, jonka jälkeen laite on toimintakykyinen. Tällainen on käytössä Microsoftin Windows 98 -käyttöjärjestelmässä. Kohdassa 6.2. hahmotellaan itsekonfiguroituvuusratkaisua tässä työssä esiteltävään järjestelmään.

5. LON-VERKON AUTOMAATTISEN KONFIGUROINTITUEN PROTOTYYPPI

5.1 Taustatiedot

Tässä esitellään keskitetty automaattinen konfigurointiratkaisu olemassa olevaan kenttäväyläjärjestelmään. VTT Elektroniikassa toteutettiin vuosina 1997–1998 Dynamoniminen tutkimusprojekti, jossa pilotoitiin mm. verkkokonfiguraattori ABB Transmit Oy:lle. Projektissa toteutettiin verkkokonfiguraattorityökalu, jolla tehdään automaattista konfigurointia LON-väylän laitteille, joita käytetään ABB:n sähkönjakeluverkon suojaus- ja ohjausjärjestelmissä. Lähteessä [19] on kuvattu konfiguraattorin spesifikaatiot, jota on hyödynnetty ja edelleen kehitetty konfiguraattorin suunnittelussa.

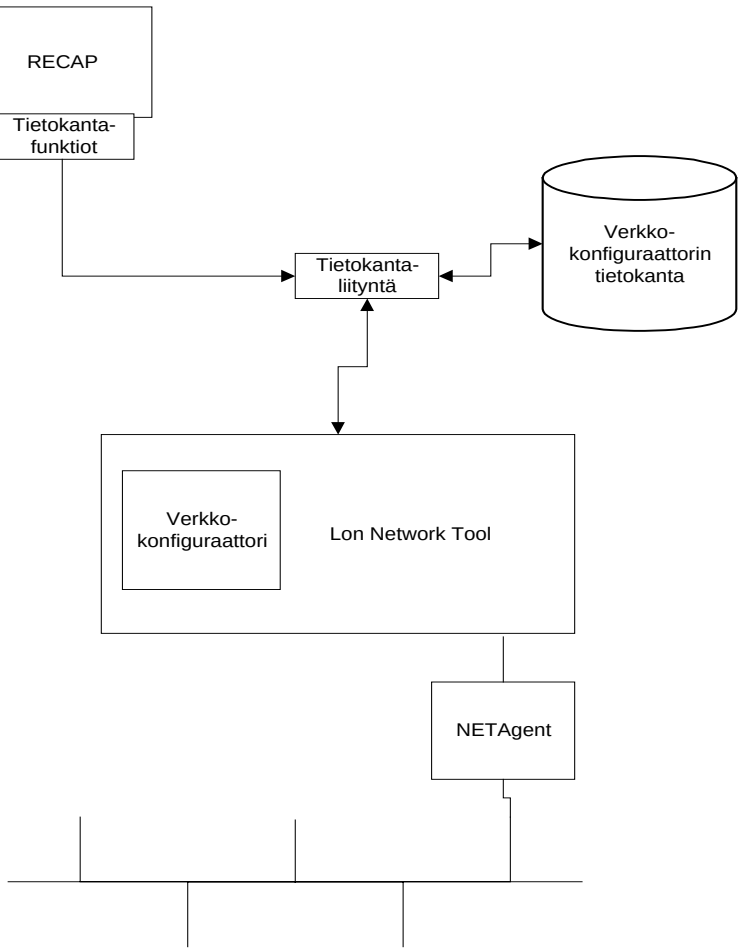
Releiden ja releitä käyttävien järjestelmien suunnittelua tuetaan eri käyttäjäryhmille tarkoitettujen työkaluohjelmistopakettien avulla. Työkalupaketteja ovat mm. CAP 501 (Relay Setting Toolbox), CAP 505 (Relay Product Engineering Toolbox) ja LNT 505. Pilotissa suunniteltu verkkokonfiguraattorityökalu tulee sisältymään LNT 505 -työkalupakettiin.

Suojareleiden hajauttaminen perustuu LON-väyläratkaisuun. Tällä hetkellä toiminnot ja alijärjestelmät kuvataan IEC 1131-3 -yhteensopivalla notaatiolla ala-asema- tai laitekohtaisesti. Jokaisen laitteen sisältö ladataan kohdesolmuun, jossa se käännetään suoritettavaksi koodiksi. LON-verkkoon laitteet näkyvät muuttujina, jotka ovat johdettavissa IEC 1131 -kuvausten rajapinnoista. Tällä hetkellä laitekohtaisten IEC 1131 -toimintokuvausten rajapintojen kytkeminen toisiin verkon laitteisiin on tehtävä manuaalisesti LNT-työkalua käyttäen. Ongelmaksi tässä lähestymistavassa on muodostunut verkkokonfiguraation hallinta LON-väylässä. Tavoittena on kehittää automatiikkaa verkon konfiguroinnin hallintaan lisäämällä nykyisen LNT-työkaluun automaattisen LON-verkon konfiguroinnin tuki.

5.2 Ratkaisumalli

Kuvassa 19 on ratkaisumallin arkkitehtuuri. Ratkaisu perustuu palvelin-asiakas-arkkitehtuuriin (client-server). Suunnittelutyökalut (RECAP ja LNT) ovat asiakkaita, jotka käyttävät konfigurointitietokantaa (palvelinta) hyväkseen. Verkkokonfiguraattori on sisällytetty LNT-työkaluun, eli kokonaan uutta työkalua ei näin tarvita ja kaikki valmiina olevat LNT-työkalun palvelut ovat käytettävissä. Olemassa olevia työkaluja käytetään edelleen hyvin samalla tavalla kuin ennenkin; ainoastaan tietokantaliityntä aiheuttaa muutoksia työkalujen käyttöön.

Konfiguroinnin automatisointi tarkoittaa sitä, että verkkokonfiguraattori osaa liittää oikeat laitteet yhteen tietokannassa olevan suunnitteluinformaation perusteella. Tämän aikaansaamiseksi verkkokonfiguraattori sisältää päättelysääntöjä. Verkkokonfiguraattorin päätehtävä on muodostaa LON-solmussa olevan Neuron-piirin kaksi taulua, jotta LON-verkko osaisi liittyä muihin solmuihin oikealla tavalla.



Kuva 19. Uudistettu suunnitteluympäristö verkkokonfiguraattorilla täydennettynä.

5.2.1 Konfigurointitiedon tuottaminen

Projekteissa syntyneet laitekuvaukset tallennetaan kirjastoon (Product Library kuvassa 18). Relejärjestelmien suunnittelussa hyödynnetään tätä kirjastoa. Laitekomponenttien suunnitteluvaiheessa tuotetaan liityntäkuvaus, joka sisältää oletusarvoiset nimet ja mahdollisesti vakioitujen liityntöjen kytkentäkuvaukset. Laitetyyppisuunnittelussa

käytetään vakiokytkentöjä, mikäli se on mahdollista, ja/tai I/O-liitynnöille määritellään niiden oletusarvoinen kommunikointitapa. Laitesuunnitteluvaiheessa tuotetaan liityntäkuvaus, joka sisältää seuraavat määrittelyt:

- DeviceType (laitetyyppi esim. REF 541)
- DefaultNetworkVariableName.

Laitetyyppien kytkentäkuvausten perusteella muodostuu järjestelmän rakenne. Järjestelmäsuunnittelussa kirjastokomponenttien (DeviceType) liitynnät (I/O-nimet) muunnetaan järjestelmäkohtaisiksi nimiksi. Järjestelmäkohtainen nimi yksilöi I/O:n kytkentätavan.

Käytössä olevista laitteista konfigurointitieto on mahdollista lukea LNT-työkalulla käyttäen apuna NetAgent-ohjelmaa. Tällöin voidaan muokata laitetietoja ja tallentaa ne sen jälkeen konfigurointitietokantaan (Network Configurator Database).

Konfiguroinnin automatisoimiseksi tarvitsee suorittaa siis seuraavat tehtävät:

- tietokannan suunnittelu (taulujen määrittely)
- tietokantaliityntä eri työkaluista, muutokset työkaluihin
- muutokset LNT-työkaluun
- päättelysääntöjen suunnittelu ja toteuttaminen.

5.3 Verkkokonfiguraattori

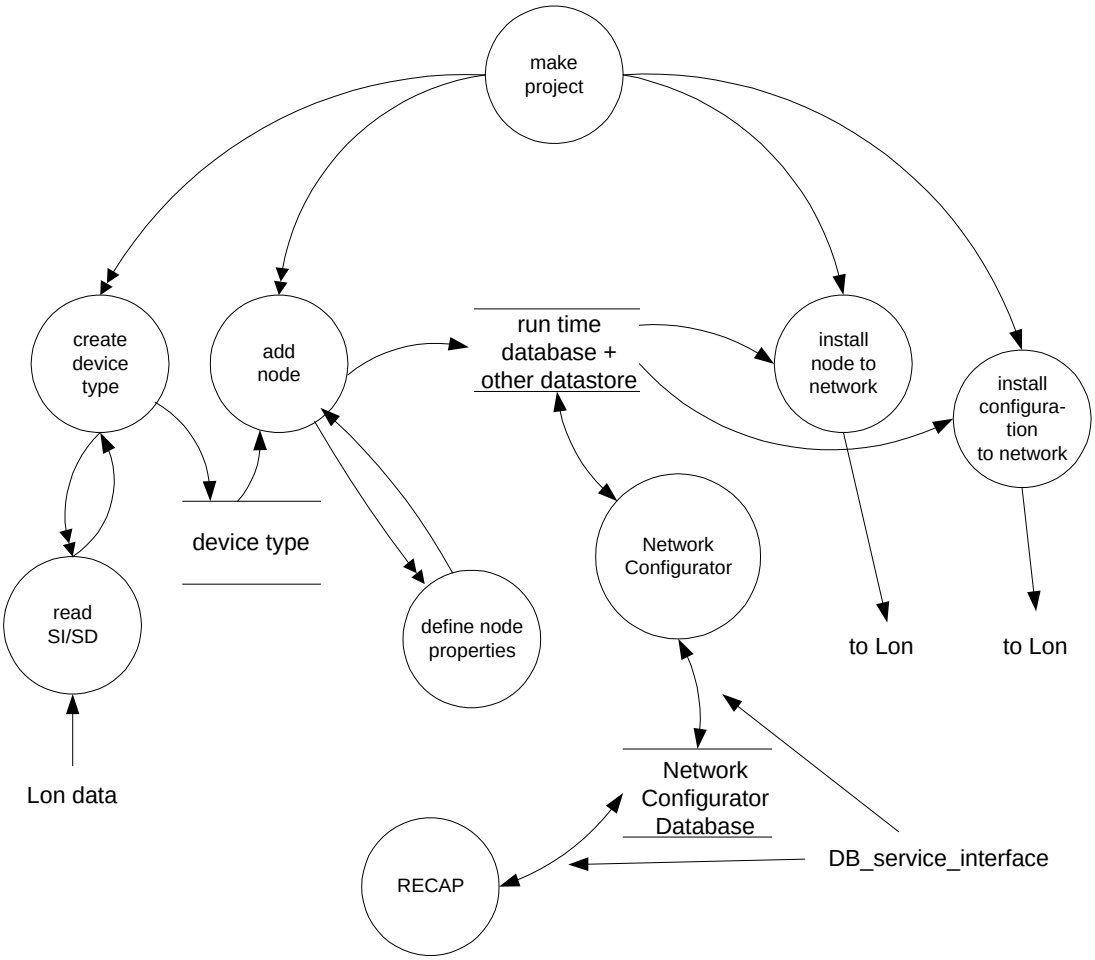
5.3.1 Verkkokonfiguraattorin tehtävät

Verkkokonfiguraattorin päätehtävä on liittää laitteet toisiinsa ja saada näin aikaan toimiva verkko. Toimiva verkko saadaan aikaiseksi

- määrittelemällä verkon sisältämät laitteet,
- hakemalla laitteiden verkkokonfigurointitiedot tietokannasta,
- lataamalla toiminnot laitteille ja
- kytkemällä laitteiden toimintojen tai alijärjestelmien tulot ja lähdöt toisiinsa.

Laitteiden väliset liitynnät toteutetaan käytännössä verkkomuuttujilla, jotka sitovat kaksi tai useampia muuttujia yhteen. Verkkokonfiguraattorin tehtäväksi jää solmun sisäisten taulujen luominen (neuronpiirin osoite- ja verkkomuuttujien konfiguraatiotaulu). Näissä kahdessa taulussa sijaitsee laitteiden välinen kytkennällisyystieto.

Kuvassa 20 on esitetty uudistettu konfigurointiproseduuri SA/SD-kaaviona. Proseduuri on muuten samanlainen kuin ennenkin (katso kuva 16), mutta verkkomuuttujien määrittely on jätetty verkkokonfiguraattorin tehtäväksi. Verkkokonfiguraattori käyttää laitteiden liittämisessä toisiinsa päättelysääntöjä, jotka esitellään kohdassa 5.3.4.



Kuva 20. Uudistettu konfigurointiproseduuri.

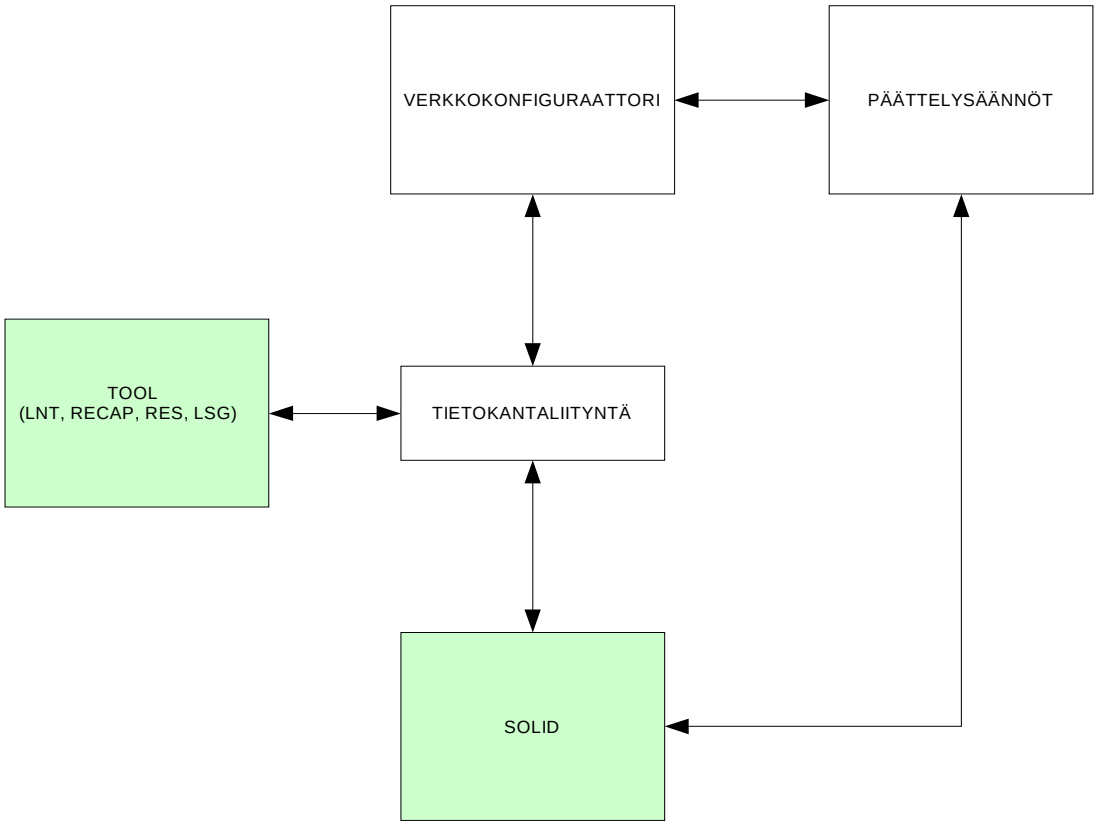
Verkkokonfiguraattorin päättelysäännöt on suunniteltu ja toteutettu LON-protokollan sääntöjen ja attribuuttien sekä sovelluskohtaisten rajoitusten perusteella. Sovelluksessa ja LON-protokollassa käytettävien attribuuttien merkitystä selitetään taulukossa 4.

Taulukko 4. Sovelluksessa ja LON-protokollassa käytettävät attribuutit.

Attribuutti	Selitys
Subnet	Aliverkon numero (0-255)
Node	Solmun numero (0-127)
Device type	Laitteen tyyppi (REF,LSG…)
Extended address	Eräissä laitteissa voi olla 255 osoitetta (norm. 15)
Address entries	Osoitteiden lukumäärä osoitetaulussa
Location	Laitteen sijainti
Priority	Prioriteetti
Network variable name	Verkkomuuttujan nimi
Authentication	Oikeuksien tarkistaminen
Repeat timer	Toistojen väli viestinlähetyksessä (UNACK_RPT)
Retry	Uusintayritysten määrä
Receive timer	Uusintayrityksen hylkäys
Transaction timer	Uusintayritysten väli (ACKD,REQUEST/RESPONSE)
Network variable index	Verkkomuuttujan indeksi
SNVT	Standardoitu verkkomuuttujatyyppi
Direction	Signaalin suunta (input vai output)
Data length	Viestin datakentän pituus
Address index	Osoiteindeksi
Turnaround	Signaalisilmukka
Connection name	Yhteyden nimi
Selector	Valitsin. Jokaisella yhteydelle annetaan myös numero.
Address type	Osoitetyyppi (Subnet/Node, Broadcast, …)
Service	Palvelu (ACKED, UNACKED,…)
Entry data	Käytetään osoitukseen osoitetaulussa
IEC standard address	Absoluuttinen osoite
IEC standard name	LSG-laitteen kanssa käytettävä attribuutti
Project	Projektin nimi

5.3.2 Integroidun järjestelmän arkkitehtuuri

Kuvassa 21 on kuvattu prototyyppijärjestelmän ohjelmistoarkkitehtuuri korkealla tasolla. Kaupalliset komponentit ovat tummennettu. TOOLiin sisältyvät ABB Transmit OY:n käyttämät suunnittelu- ja konfigurointityökalut LNT, RECAP, RES ja LSG. Tämän työn kannalta ainoastaan LNT- ja RECAP-työkalut ovat merkittäviä. Tietokantaliityntä sisältää liityntäfunktioita ja se on toteuttu DLL-tiedostona (dynamic link library). Päättelyn aikana ja sen jälkeen tehtävissä tietokantakutsuissa ei käytetä liityntäkirjastoa, vaan se käyttää omaa liityntää, joka ei ole DLL-tiedosto. Verkkokonfiguraattorilla kuvassa 21 tarkoitetaan lähinnä käyttöliittymää, jolla ohjataan konfigurointiproseduuria. Päättelysäännöt on kuvattu erikseen, koska niillä on erilainen liityntä tietokantaan.



Kuva 21. Järjestelmän ohjelmistoarkkitehtuuri.

5.3.3 Tietokantataulut

Relaatiotietokanta muodostuu tauluista, joilla on yhteys toisiinsa. Taulujen yhteyden havainnollistamiseen käytetään ER-mallia (Entity-Relationship). Tämän sovelluksen ER-malli on esitetty liitteessä 1. Mallissa on ainoastaan taulujen perusavaimet ja yhteydet toisiin tauluihin. Taulukossa 5 käsitellään taulujen välisiä viiteavaimia.

Taulut luodaan tietokantaan verkkokonfiguraattorilla. Projektit tunnistetaan *project*-attribuutin avulla. Tietokantaan pääsy hoidetaan *server*, *uid* (user id) ja *pwd*- (password) -attribuuttien avulla (ODBC:n ominaisuus).

Tietokantataulujen sisältö

Alla olevissa tauluissa on relaatiomalli konfigurointiin liittyvistä tietokantatauluista. Taulujen yläosaan on merkitty, mistä taulu on saanut eri attribuuttinsa (informaation lähde). DEVICE tarkoittaa, että tieto on itse laitteesta saatua. DEFAULT on oletusarvo tai vakioita, jotka eivät muutu. USER on verkon suunnittelijan luomaa informaatiota. INFERRED tarkoittaa päättelyn tuloksena syntyneitä attribuutteja. Taulut luodaan seuraavassa järjestyksessä taulujen välisten viittausten takia. Attribuuttien lyhenteet ja tietotyypit ovat liitteessä 2.

1. ND_TABLE (Node_Table)

DEVICE					
subnet	node	dev_type	ext_add	add_ent	location

1. NAME_CON (Name_Conversion)

DEFAULT	
iec_stan_add	nv_idx

1. DEV_TABLE (Device _Table)

DEVICE					
subnet	node	NV_index	SNVT	dir	datalen

2. NV_TABLE (NetworkVariable_Table)

USER	DEVICE			USER		DEFAULT			
project	subnet	node	prio	nv_name	auth	repeat_tmr	retry	rec_tmr	trns_tmr

DEVICE			
nvidx	snvt	dir	datalen

INFERRED					
add_idx	tround	con_name	sel	add_type	ser

1. ADD_TABLE (Address_Table)

INFERRED			
subnet	node	add_idx	entry_data

2. ADD_CON (Address_Conversion)

DEFAULT	
iec_stand_name	lsg_con_string

Seuraavassa kerrotaan lyhyesti taulujen tarkoitus ja taulujen väliset viittaukset.

Taulukko 5. Verkkokonfiguraattorin tietokantataulut ja niiden yhteydet.

Taulu	Tarkoitus ja yhteydet
ND_TABLE (Laitetaulu)	Laitetaulussa on jokaisesta laitteesta sen perustiedot. Kun uusi laite luodaan, sen tiedot syötetään ensimmäiseksi juuri laitetauluun. Muissa tauluissa ei voi olla laitetta, jota tässä taulussa ei ole.
NAME_CON (konversio)	NAME_CON on vakiotaulu, millä saadaan valittua oikea verkkomuuttujaindeksi, kun syötteenä on absoluuttinen osoite (iec_stand_add). Absoluuttinen osoite määritellään RECAP-työkalussa.
DEV_TABLE (Laitteen sisäinen konfigurointitaulu)	Tässä taulussa on jokaisen laitteen ns. datalehti eli tiedot laitteen sisäisestä konfiguroinnista. Tästä taulusta on viittaus laitetauluun, jossa täytyy olla kyseisen laitteen perustiedot. Tämän taulun arvot ovat SI/SD-informaatiota (Self Identification/Self Documentation), ja ne voidaan ladata laitteelta tietokantaan.
NV_TABLE (Kytkennällisyys-taulu)	Kytkennällisyystaulussa on tiedot laitteiden välisistä kytkennöistä. Taulusta on viittaus laitetauluun ja laitteiden sisäiseen konfigurointitauluun.
ADD_TABLE (Osoitetaulu)	Tässä taulussa pidetään yllä tietoa laitteesta lähtevistä signaaleista. Taulusta viitataan laitetauluun.
ADD_CON (konversio)	Tämä taulu on konversiotaulu, jossa muutetaan RECAP-työkalussa määritelty LSG-laitteen parametri LON-spesifiseksi. Tähän tauluun ei viitata mistään eikä tästä taulusta viitata mihinkään.

5.3.4 Verkkokonfiguraattorin päättelysäännöt

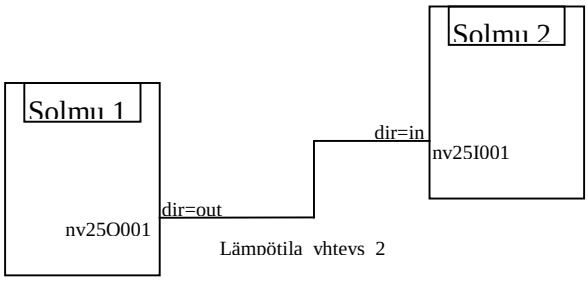
Verkkokonfiguraattorin tulee päätellä seuraavat LON-spesifiset attribuutit: *connection name* (yhteyden nimi), *selector* (valitsin), *address type* (osoitetyyppi), *address index* (osoiteindeksi), *turnaround* (signaalisilmukka) ja *service* (palvelutyyppi). Seuraavassa selitetään säännöt siihen, mitenkukin attribuutti muodostetaan. Sääntöjen 1, 2 ja 3 avulla määritellään solmujen väliset yhteydet ja säännöllä 4 määritellään solmun sisäinen viittaus kohdesolmuun tai kohdesolmuihin. Kytkentöjen erikoisominaisuudet määritellään säännöissä 5 ja 6. Taulukko attribuuttien lyhenteistä on liitteessä 2. Päättelyt tehdään pääsääntöisesti esitetyssä järjestyksessä (signaalisilmukka päätellään yhteyden nimen päättelyn yhteydessä).

1. Yhteyden nimi

Yhteyden nimellä nimetään laitteiden välinen yhteys. Yhteyden nimen määrääminen on erittäin olennainen, sillä attribuutti sitoo varsinaisesti laitteet. Yhteyden nimi päätellään signaalin verkkomuuttujan nimen ja suunnan perusteella. Päättelysääntö: Jos kahdella

signaalilla on sama verkkomuuttujan nimi ja suunnat ovat erilaiset, niin molempien solmujen signaalien yhteyden nimeksi annetaan sama arvo. Kuvassa 22 on havainnollistettu yhteyden nimen muodostumista (attribuutin arvoksi on annettu Lämpötila_yhteys_2).

Yhteyden nimi on yksilöllinen, eli kahdella eri yhteydellä ei voi olla sama nimeä. Nimi on yhteneväinen valitsimen kanssa, mutta sitä käytetään siksi, että suunnittelija ymmärtää paremmin kuvaavan nimen kuin numeron.



Kuva 22. Connection name -attribuutin päättely.

Päättely muuttuu hiukan, jos kyseessä on broadcast-tyyppinen lähetys: sama yhteyden nimi annetaan kaikille saman verkkomuuttujan nimen omaaville signaaleille.

2. Valitsin

Valitsinta ei varsinaisesti päätellä, vaan jokaiselle yhteydelle annetaan yksilöllinen arvo, joka on yhteneväinen yhteyden nimen kanssa. Päättelysääntö: Kaikille samaan verkko-muuttujayhteyteen kuuluville annetaan sama valitsimen arvo. Arvot annetaan järjestyk-sessä ykkösestä lähtien, ja samaa lukua ei anneta toista kertaa.

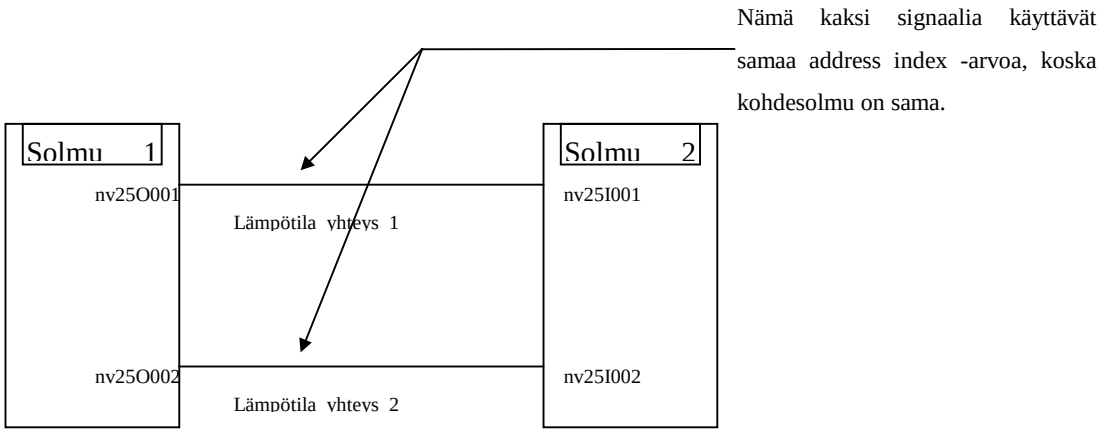
3. Osoitetyyppi

Päättelysäännöt: Jos yhteydellä on vain yksi sisääntulo-arvo (input), niin osoitetyyppi on subnet/node “01”. Jos yhteydessä on useampi sisääntulo-arvo, osoitetyyppi on broadcast “03”.

4. Osoitetyyppi

Osoiteindeksiä käytetään lähtevien signaalien ohjaukseen. Koska solmukohtaisessa osoitetaulussa on vain 15 osoitetta, eri yhteydet voivat joutua käyttämään samoja

osoiteindeksejä. Tämä tietenkin edellyttää sitä, että kahden solmun välillä on useita verkkomuuttujayhteyksiä. Verkkokonfiguraattorin tehtäväksi jää osoiteindeksin päättely. Päättelysääntö: Käydään läpi lähteviä signaaleja ja niiden kohdesolmuja. Tarkistetaan, onko osoitetaulussa jo tarvittava kohdesolmu (osoitetyypin tulee olla sama: joko subnet/node tai broadcast). Kuvassa 23 on havainnollistava esimerkki osoiteindeksin käytöstä.



Kuva 23. Address index -attribuutin päättely.

Verkkomuuttujataulussa tulisi olla taulukon 6 attribuutit, jotta osoiteindeksin uudelleenkäyttö olisi mahdollista.

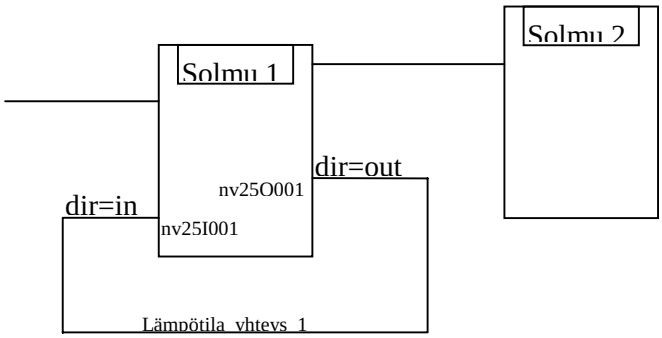
Taulukko 6. Tarvittavat arvot verkkomuuttujataulussa, jotta osoiteindeksin uudelleenkäyttö olisi mahdollista.

subnet	node	direction	address type	con_name
1	1	0	subnet/node	sininen
1	1	0	subnet/node	valkoinen
1	2	1	subnet/node	sininen
1	2	1	subnet/node	valkoinen

Liitteessä 3 on selvitelty verkkomuuttujien konfigurointi- ja osoitetaulussa (huom. neuronpiirin osoitetaulu) esiintyvien attribuuttien rakennetta.

5. Signaalisilmukka

Jos yhteyden sisään- ja ulostulo sijaitsevat samassa solmussa, kyseisellä yhteydellä on signaalisilmukka-ominaisuus. Tämä ominaisuus tarkistetaan silloin, kun kytkentää (yhteyden nimeä) ollaan päättelemässä. Päättelysääntö: Ensinnäkin osoitetyypin tulee olla subnet/node eli “01”. Jos saman yhteyden nimen omaavat signaalit liittyvät samaan solmuun, yhteydellä on signaalisilmukka-ominaisuus. Kuvassa 24 on esimerkki yhteyden signaalisilmukka-ominaisuudesta.



Kuva 24. Signaalisilmukan päättely.

6. Palvelutyyppi

Palvelutyyppejä on neljä kappaletta, ACKED, UNACKED, UNACKED REPEATED ja REQUEST/RESPONSE, joista viimeistä ei kuitenkaan koskaan käytetä käytännön sovelluksissa. Päättelyssä käytetään seuraavia sääntöjä:

Jos osoitetyyppi on subnet/node, käytetään ACKED-palvelua (oletusarvo).

Jos kysymyksessä on LSG-laite, käytetään UNACKED REPEATED -palvelua. (jos RETRY on kaksi).

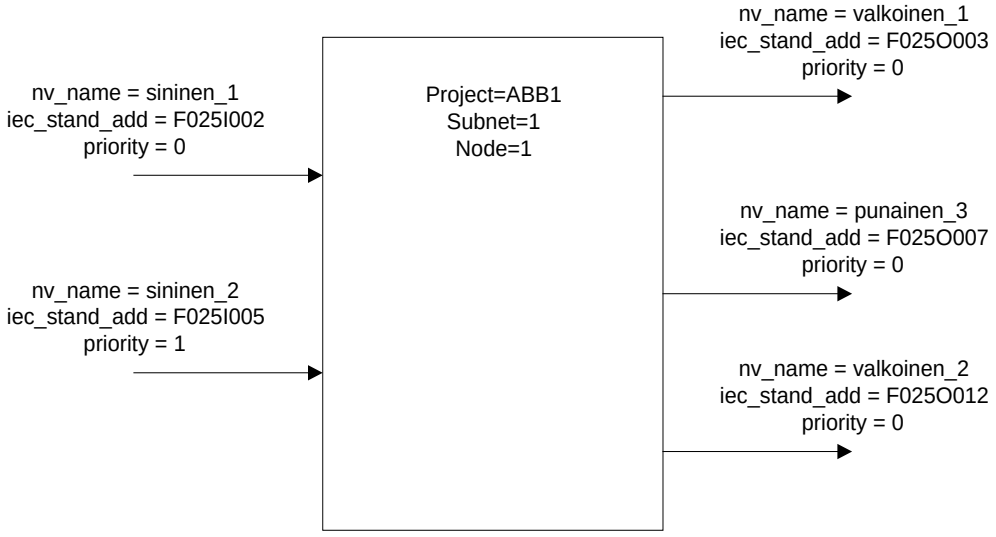
Jos lähetys on broadcast-tyyppinen, käytetään UNACKED REPEATED -palvelua.

Näiden päättelyiden jälkeen kysytään suunnittelijalta, haluaako hän itse valita palvelutyypin (semiautomaattinen konfigurointi). Valinnalla kumotaan aikaisemmin tehty päätös tyypistä.

5.3.5 Tietokantaliityntä

Tietokantaliityntä sisältää funktioita, joita eri työkalut kutsuvat joko kirjoittaakseen tai lukeakseen tietoja kannasta. Liityntäfunktiot ovat hiukan erilaiset eri työkaluille, koska toimitettavat operaatiot ovat erilaisia. Tietokantaliityntä on toteutettu ODBC-rajapinnan avulla, ja sen funktiot ont sisällytetty DLL-tiedostoon. Tämä DLL-tiedosto tulisi linkittää suunnittelu- ja konfigurointityökaluihin, jotta ne pystyisivät käyttämään tietokantaliityntää.

Olennainen asia verkon konfiguroinnin kannalta on liityntätietojen syöttö tietokantaan. Laitteiden liityntätiedot syötetään kantaan RECAP-työkalusta siihen tarkoitetulla funktiokutsulla. RECAPilla tehtävää suunnittelutyötä on pyritty vähentämään päättelysääntöjen kustannuksella. Työkalussa tarvitsee määrätä seuraavat attribuutit: nv_name, iec_stand_add ja priority. Jos kyseessä on LSG-laite, myös iec_stand_name -attribuutti tulee määrätä. Nv_name osaa yhdistää oikeat signaalit toisiinsa, ja iec_stand_add hakee yhteydelle oikean indeksin konversiotaulusta (iec_stand_name tekee saman LSG-laitteelle). Kuvassa 25 on havainnollistettu RECAPissa tehtävää suunnittelua.



Kuva 25. Laitteen liityntätiedot, jotka tuotetaan RECAP-työkalulla.

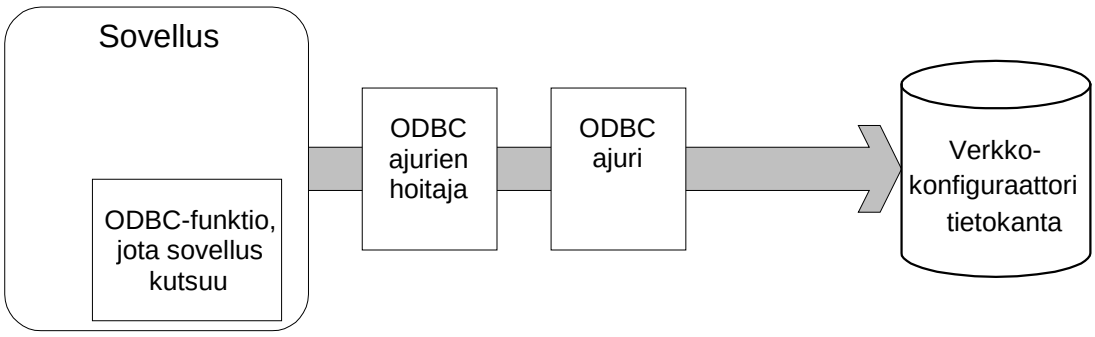
Työkaluista on mahdollisuus hakea tietoja konfiguroinneista eri valintafunktioilla. Valintoja voi tehdä taulu-, projekti- tai laitekohtaisesti. Taulujen luomiseen sekä laitteiden syöttöön, päivityksiin ja poistoihin on toteutettu omat funktionsa. Tärkeätä konfigurointiproseduurissa on tehdä tietokantatoimenpiteet oikeassa järjestyksessä, jotta verkon konfigurointi onnistuu.

5.4 Tietokanta konfiguroinnin apuna

5.4.1 ODBC-rajapinta

Tiedonsiirto sovelluksesta tietokantaan tapahtuu ODBC-rajapinnan avulla. Open DataBase Connectivity on Microsoftin kehittämä rajapinta. ODBC toimii vain Windows-sovelluksissa ja ohjelmointikielenä käytetään C-kieltä. Sovellus kutsuu funktiota, joka suorittaa varsinaiset tietokantatoimenpiteet. Kuvassa 26 on esitetty malli siitä, miten tieto siirtyy sovelluksesta tietokantaan. ODBC-ajurien hoitaja valitsee oikean ajurin, koska ajureita voi olla useita. Asiakastyöasemaan täytyy olla asennettuna ODBC-ajuri ennenkuin tietokantayhteys voidaan muodostaa. Sovelluksessa täytyy myös määrätä, mitä ajuria käytetään. Samassa työasemassa voi olla useita tietokanta-ajureita.

Yhteyden muodostamiseksi tietokannan tulee olla auki ja sovelluksella täytyy olla tiedossa tietokannan nimi, käyttäjätunnus ja salasana. Jos yksikin näistä tiedoista puuttuu, yhteyttä ei voida muodostaa. Kun yhteys on luotu, voidaan suorittaa varsinaiset SQL-komennot. SQL-kysely lähetetään merkkijonona ja parametrit valitaan sovitulla tavalla. Jokainen kysely palauttaa arvon. Jos kyselyt eivät jostain syystä onnistuneet, tulostetaan virheilmoitukset. Kun kyselyt on suoritettu ja mahdolliset virheet huomattu, voidaan tietokantayhteys sulkea [35].



Kuva 26. ODBC-kutsun rakenne.

5.4.2 Solid-relaatiotietokanta ja oliotietokannat

Relaatiokannat ovat käteviä verrattuna tiedostoihin, sillä relaatiokannoissa tieto on järjestetty ja määrämuodossa. Tiedonhaku kannoista on mutkatonta ja nopeaa. Varsinkin tämäntyyppisessä sovelluksessa, missä on paljon attribuutteja ja niiden välillä yhteyksiä, relaatiokanta on ehdottomasti paras vaihtoehto.

Oliotietokanta on tiedonhallintajärjestelmä, joka integroi tietokannan ja olio-suuntautuneen ohjelmointikielen ominaisuudet ja mahdollisuudet. Käytännössä oliotietokanta tekee tietokantaobjekteista ohjelmointikielen objekteja. Oliotietokantojen ohjelmoinnissa käytetään mm. C++-, Smalltalk- ja Java-ohjelmointikieliä [34].

Konfigurointitietokannaksi valittiin relaatiotietokanta Solid, koska sen oletettiin olevan helppokäyttöinen, skaalattava ja luotettava. Solid Server on relaatiokanta, joka on skaalattavissa pienistä sovelluksista isoihin, joten se ei ollut liian raskas konfiguraattorin tietokannaksi. Vaatimuksena tietokantajärjestelmälle oli myös ODBC-liitynnän käytön mahdollisuus. Solid-tietokannalle löytyy ODBC-ajuri ja Solid SQL API-ohjelmointia havainnollistavia esimerkkejä [33].

Tässä sovelluksessa Solid-tietokanta toimi melko moitteettomasti. Ennen kaikkea sovellusta oli helppo käyttää. Asennus oli nopea ja eri ohjelmien käyttö oli helppoa. Satunnaisia käyttöhäiriöitä kuitenkin ilmeni. Editorissa suoritetut SQL-komennot eivät aina toimineet ja kun SQLEditor-sovellus oli ollut pitkään auki, sen toiminta pysähtyi. Käyttötuki toimi erinomaisesti ja kannan suorituskykykin oli kohtuullinen.

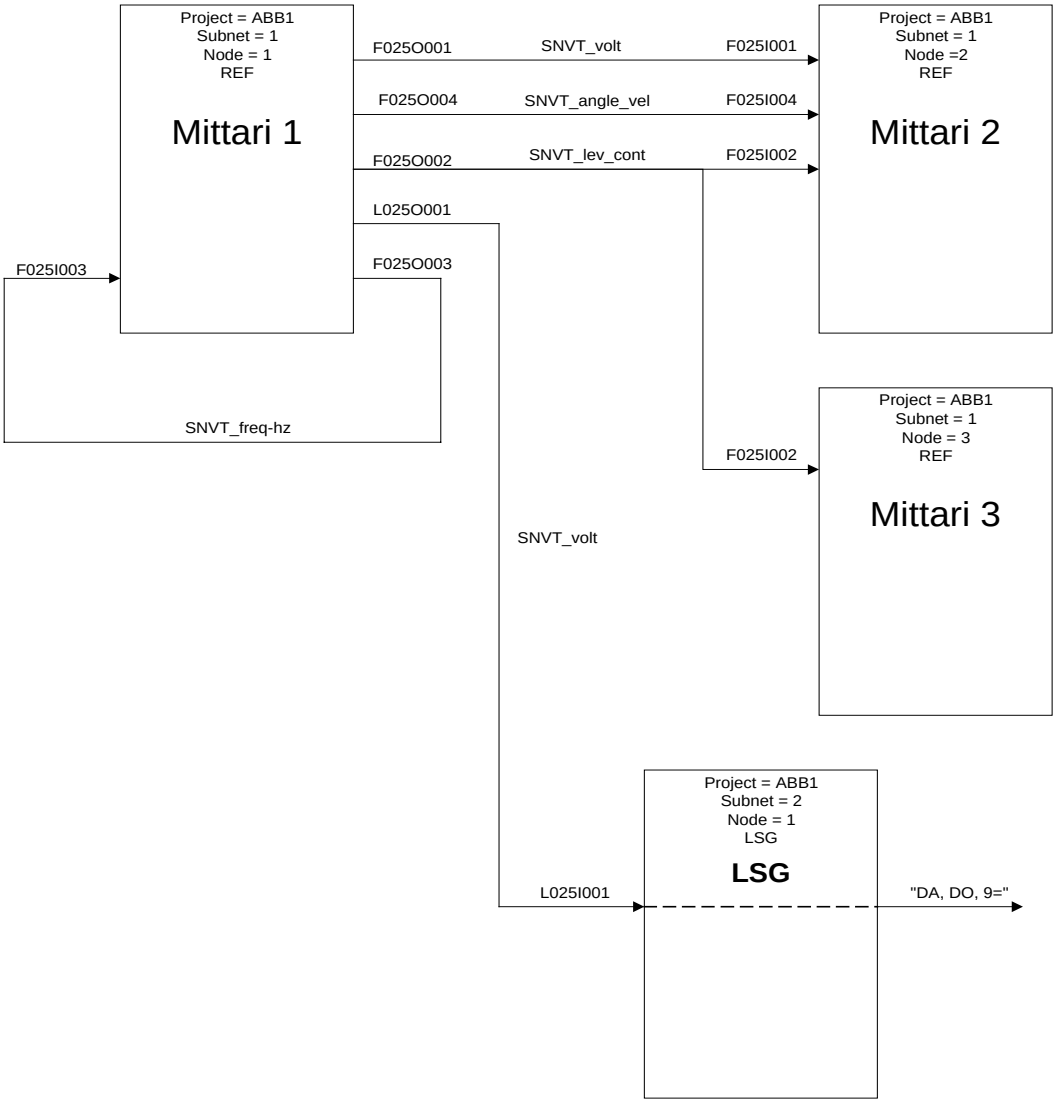
Verkkokonfiguraattorin tietokantana olisi voitu käyttää myös oliotietokantaa. Vaihto onnistuu helposti, jos OODB:hen on olemassa ODBC-liityntä.

5.5 Ratkaisun käytettävyys LON-väylässä

5.5.1 Testiverkko

Testauksessa pyrittiin saamaan selville se, suorittaako verkkokonfiguraattori tarvittavat päättelytoiminnot oikealla tavalla. Ratkaisun toimivuus testattiin esimerkillä, joka on kuvassa 27. Esimerkissä on neljän laitteen verkko, jossa on erityyppisiä yhteyksiä, ja laitetyyppejä on kahdenlaisia (REF ja LSG). Testauksen apuna käytettiin windows-käyttöliittymää, joka esitellään kohdassa 5.5.3.

Todellisissa sovelluksissa verkot voivat olla paljon suurempia, mutta tässä tapauksessa esim. usean kymmenen laitteen testaus ei palvele tarkoitusta (prototyyppi). Testiverkko on kuitenkin suunniteltu siten, että kaikkia päättelysääntöjä tulee testattua.



Kuva 27. Testiverkko.

5.5.2 Testauksen tulokset ja arvioita työstä

Testiverkon konfigurointi onnistui virheittä mutta suorituskyky ei ollut tyydyttävä. Yhden yhteyden lisääminen edellä kuvattuun testiverkkoon onnistui myös virheittä. Pullonkaulaksi osoittautui tietokantaliitynnän hitaus. Syynä on koodin optimoimattomuus. Silmukat olisi voinut rakentaa eri tavalla ja hakuproseduuria olisi voinut siirtää enemmän tietokannan tehtäväksi. Tämä tarkoittaa sitä, että SQL-lauseet olisi voinut suunnitella järkevämmiksi. Tällä hetkellä tehdään liikaa työtä tietokantaliitynnässä ja päättelysäännöissä, kun osan työstä olisi voinut hoitaa tietokannan valmiilla palveluilla. Tarkoitus oli kuitenkin ensisijaisesti tehdä toimiva

ohjelma automaattisen konfiguroinnin suorittamiseksi. Verkkokonfiguraattori siis toimii oikein mutta hitaasti. Taulukossa 9 luetellaan sovelluksen eri toimintoihin kuluva aika.

Taulukko 9. Verkkokonfiguraattorin suorituskyky.

Toiminta	Aika [ms]
Kannan aukaisu	500
Kannan sulkeminen	350
Kursorin asetukset	260
Select: 4 riviä, 7 attribuuttia	3 400
Select: 11 riviä, 21 attribuuttia	8 200
Select all	17 000
Create	280
Vakiotaulujen täyttö	2 000
Insert	680
Update (ND_TABLE)	720
Update (NV_TABLE)	2 400
Projektin tuhoaminen	200
Max_sel -funktio	1 000
Update_after_rules	1 600
Koko päättelyproseduuri	66 000

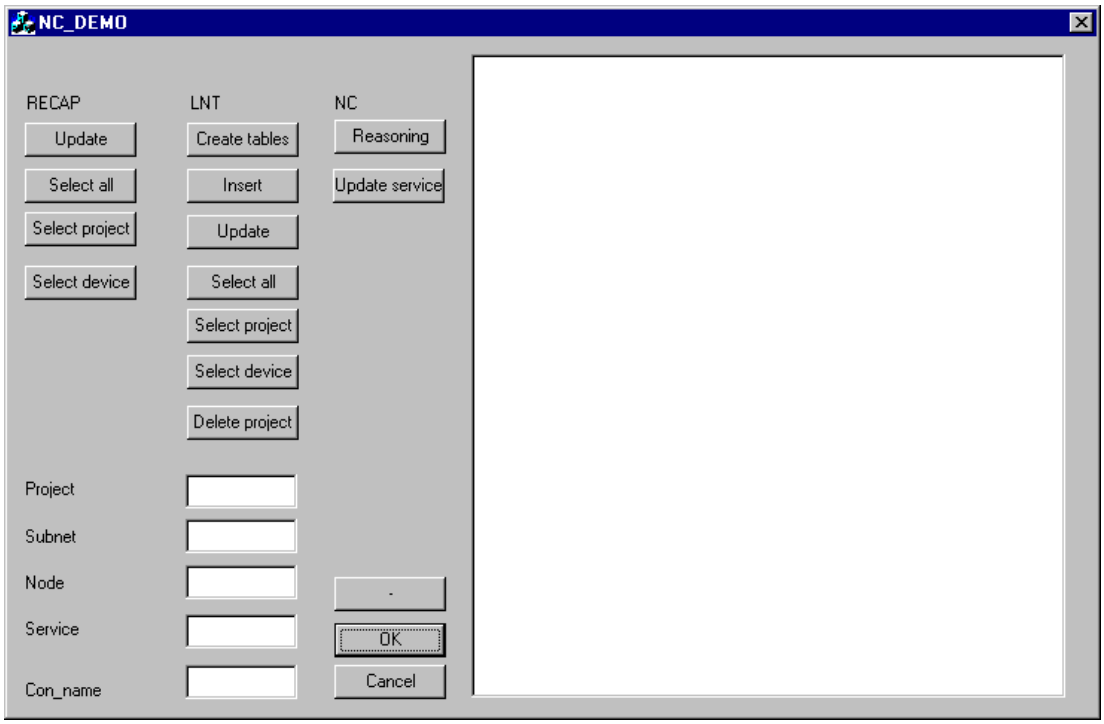
Toteutusvaiheen alussa ongelmia aiheutti ODBC-liityntä, josta ei ollut saatavilla kunnollista ohjelmointiopasta. Vaikeuksia testauksessa ja virheenilmaisussa aiheutti SQLError-funktion toimimattomuus käyttämässämme Borlandin C++-kääntäjässä. Myös kursoreiden käytössä valinnan yhteydessä ilmeni virhetilanteita. Virhetilanteet poistuivat, kun kääntäjäksi otettiin Microsoftin Visual C++. Omat ongelmansa aiheuttivat ODBC-ajurien eri versiot ja niiden yhteensopimattomuus järjestelmään.

Pilotin vaatimukset tulivat täytettyä toteutetulla ratkaisulla, koska päättelyt ja tietokanta-toimenpiteet toimivat oikein. Projektin alussa tarkoituksena oli sisällyttää verkkokonfiguraattori LNT-työkaluun, mutta se todettiin liian vaativaksi ja riskialttiiksi. LNT-työkalu on toteutettu visual basic -ohjelmointikielellä ja lähdekoodia ei ollut kommentoitu, joten demonstraatio päätettiin toteuttaa aivan erillään todellisista työkaluista. Todellinen kuva verkkokonfiguraattorin toiminnasta saataisiin, jos se integroitaisiin nykyiseen LNT-työkaluun ja suunnittelutyökaluihin tehtäisiin tietokantaliitynnän vaatimat muutokset. Pilotissa todettiin ainoastaan tietokantaliitynnän ja päättelysääntöjen toimivuus

demonstraatioympäristössä. Projektin määrittelyvaihe kesti n. 3 htkk ja suunnittelu- ja toteutusvaihe n. 6 htkk.

5.5.3 Verkkokonfiguraattoridemonstraation käyttöliittymä

Käyttöliittymällä on tarkoitus ainoastaan esitellä ja testata konfigurointia, joten se on tehty mahdollisimman yksinkertaiseksi. Käyttöliittymässä on yhdistetty kolmen eri työkalun toiminnot samaan ikkunaan, jollaisesta on esimerkki kuvassa 28. LNT:llä suoritetaan peruskonfigurointi, ja NC (Network Configurator, verkkokonfiguraattori) suorittaa automaattisen konfiguroinnin. Kun halutaan hakea tiedot tietystä projektista tai laitteesta, käyttäjä voi syöttää halutun projektin tai laitteen tunnisteet. Semi-automaattisessa konfiguroinnissa käyttäjällä on mahdollisuus itse valita palvelutyyppi, vaikka tämäkin attribuutti päätellään proseduurissa. Jokaisella yhteydellä on oma palvelutyyppi, joten lisäksi tarvitaan yhteyden nimi kohdentamaan oikea yhteys. Käyttöliittymä on toteutettu käyttämällä hyväksi Visual C++ -työkalun MFC-kirjastoja (Microsoft Foundation Classes), koska niistä oli aikaisempia käyttökokemuksia.



Kuva 28. Testauksessa käytetty käyttöliittymä.

6. RATKAISUMALLIN ARVIOINTI JA JATKOKEHITYSMAHDOLLISUUDET

6.1 Ratkaisun sopivuus muihin kenttäväyläratkaisuihin (CAN)

Ratkaisu toimi kohtuullisesti LON-väyläratkaisussa, ja kiinnostavaa onkin se, voisiko kyseinen ratkaisu toimia jossain muussa kenttäväyläratkaisussa. Monesti käyttäjät eivät halua olla riippuvaisia yhdestä ratkaisusta vaan haluavat sen toimivan myös muissa järjestelmissä. Vaihtoehtoiseksi kenttäväyläksi on otettu CAN-väylä. CAN-väylän tekniikkaa selviteltiin kohdassa 2.4.4.

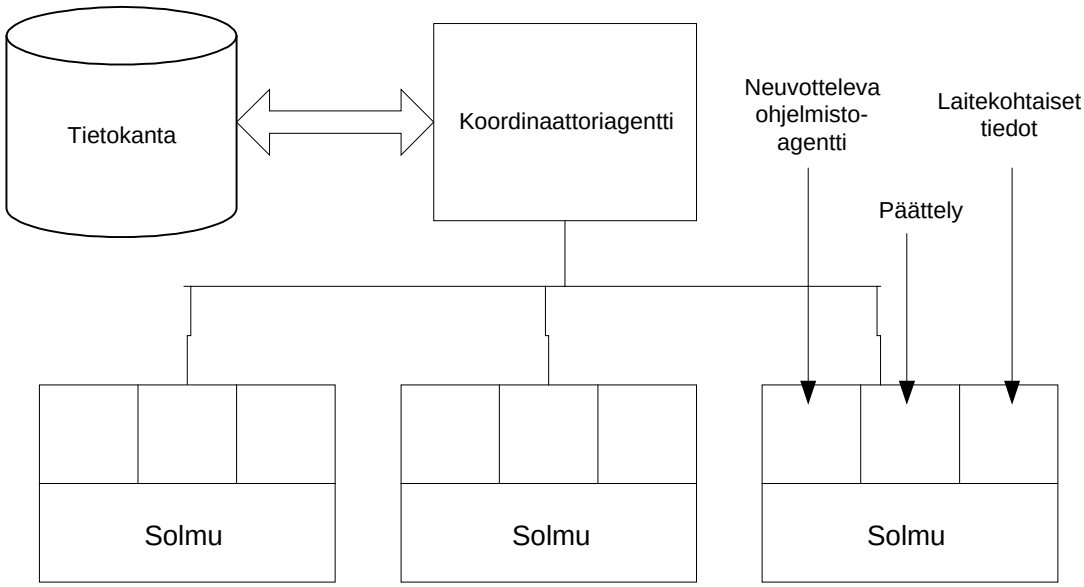
LON- ja CAN-väylät ovat erilaisia, kuten kaikki kenttäväylät. Molemmat väylät perustuvat kuitenkin lyhyiden viestien lähetykseen. Viestien lähetystapa ja formaatti ovat kuitenkin hyvin erilaisia. LON-protokollassa viesti voidaan lähetetään tietylle solmulle, kun käytetään subnet/node-osoitustapaa. Protokollassa on myös mahdollisuus lähettää viesti broadcast- tai group-tyyppisesti. CAN-protokollassa viestissä ei mainita, kenelle se on menossa, vaan solmut, jotka tarvitsevat viestin, kopioivat sen itselleen. CAN-protokollassa solmulla ei siis ole osoitetta, vaan se käyttää objektiorientoitunutta tiedonvälitystä. Erilaisia objekteja voi olla 2 032 kappaletta.

Mikäli tässä työssä esitetty ratkaisu toimisi CAN-protokollan ehdoilla, tulisi järjestelmään tehdä isoja muutoksia. Ratkaisun ohjelmistoarkkitehtuuri voisi pysyä ennallaan, mutta tietokantaan, tietokantaliityntään ja päättelysääntöihin tulisi isoja muutoksia. Koska väylien protokollan attribuutit ovat ihan erilaiset, tietokannan taulut tulisi suunnitella CAN-protokollassa toimiviksi. LON-väylän automaattisessa konfiguroinnissa käytetyt päättelysäännöt eivät toimisi CAN-väylässä, vaan ne jouduttaisiin suunnittelemaan kokonaan uudestaan. Tietokantaliityntää voisi jonkin verran hyödyntää. Ratkaisun käytettävyys riippuisi myös CAN-laitteiden ja -verkkojen suunnittelutyökaluista ja -proseduurista.

Tässä työssä esitetty ratkaisu voisi toimia CAN-väylässä, mikäli arkkitehtuurin eri komponentien sisältö muutettaisiin vastaamaan CAN-protokollan tarpeita. Perus-arkkitehtuuriin ei välttämättä tarvitse tehdä muutoksia.

6.2 Laitteiden installointi plug-in-periaatteella

Tässä kohdassa pohditaan plug-in-installointia nimenomaan esitetyn ratkaisun arkkitehtuurin uudelleenkäytön näkökulmasta. Teknisiin näkökohtiin ei oteta kantaa.

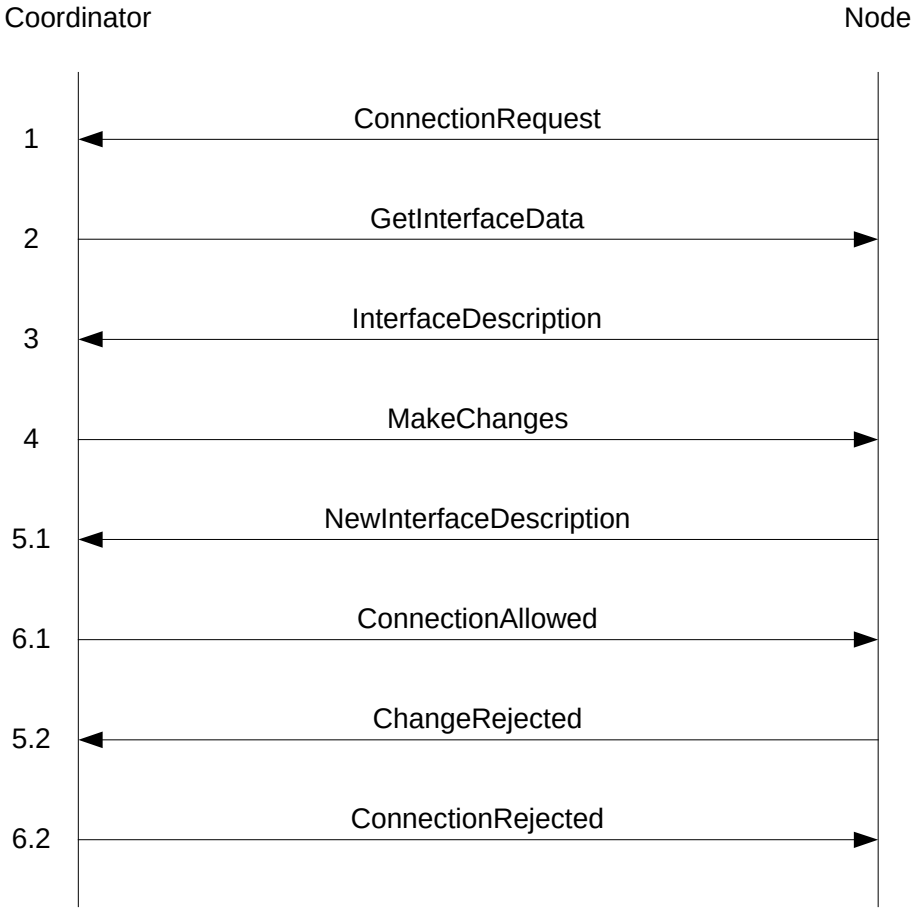


Kuva 29. Hahmotelma plug-in-installoinnista.

Mikäli automaattista konfigurointia halutaan edelleen kehittää, voisi ratkaisuna olla itsekonfiguroituvuus eli laitteiden installointi plug-in-periaatteella. Itsekonfiguroituvuutta pohdittiin kohdassa 4.4. Itsekonfiguroituvuudessa installoitava laite sisältäisi erittäin paljon älykkyyttä (prosessointikykyä) ja tietoa (muistia). Päättelysäännöt ja liityntäkuvaukset olisivat valmiina laitteessa, joka sitten osaisi itse liittyä verkkoon ja muodostaa loogiset yhteydet suunnitellusti. Kuvassa 29 on hahmoteltu plug-in-installointia. Etuna tällaisessa järjestelmässä on integrointityön väheneminen ja haittana kasvava muistin tarve solmussa.

Suunnitteluproseduuri ja ohjelmistoarkkitehtuuri olisivat hyvin erilaiset kuin tässä työssä esitetyllä ratkaisulla. Laite- ja verkkosuunnittelutyökaluihin ei välttämättä tarvitse tehdä muutoksia, vaan muutokset tehtäisiin varsinaiseen konfigurointiproseduuriin. Plug-in-installointi tarkoittaisi sitä, että laite tulee ohjelmoida, ennen kuin se fyysisesti installoidaan verkkoon. Tähän tarkoitukseen tarvitaan oma työkalu ja liitäntä laitteeseen. Tässä työssä esitellyn ratkaisun ajatusta voisi käyttää hyväksi plug-in-installoinnissa. Suunnittelutiedot talletettaisiin edelleen tietokantaan, josta ne sitten olisi eri työkalujen käytettävissä. Varsinainen konfigurointityökalu ohjelmoisi laitteen tietokannassa olevien tietojen perusteella, eli verkkokonfiguraattori ja sen päättelysäännöt korvattaisiin koordinaattorilla, joka ohjelmoisi jokaisen laitteen yksitellen.

Koordinaattorin ja laitteen välinen neuvottelu installoinin aikana on esitetty kuvassa 30. Aluksi solmu (node) ilmoittaa halukkuudesta liittyä verkkoon koordinaattorille. Koordinaattori pyytää tämän jälkeen solmulta liityntäkuvausta, johon solmu sitten vastaa lähettämällä kuvauksen (InterfaceDescription). Mikäli koordinaattori hyväksyy liitynnät, se lähettää ConnectionAllowed-viestin solmulle ja tällöin solmu on liitetty verkkoon. Mikäli solmun liityntäkuvaus on puuttellinen tai väärä, koordinaattori antaa solmulle mahdollisuuden tehdä muutoksia kuvaukseen (MakeChanges). Tämän jälkeen solmu joko tekee muutokset (NewInterfaceDescription) tai hylkää ehdotetut muutokset (ChangeRejected). Jos ehdotukset hylätään, koordinaattori kuittaa installoinnin epäonnistuneeksi.



Kuva 30. MSC-kaavio (Message Sequence Chart) koordinaattorin ja solmun välisestä neuvottelusta.

7. YHTEENVETO

Työssä selvitettiin, kuinka hajautettujen järjestelmien konfigurointia voidaan tehostaa, ja toteutettiin automaattisen konfiguroinnin suorittava verkkokonfiguraattorin prototyyppi.

Verkkokonfiguraattori integroi projekti- ja laitetietokannat yhdeksi tietokannaksi ja tuottaa päättelyn avulla puuttuvat attribuutit verkon liityntöjen aikaansaamiseksi. Suunnittelu- ja konfigurointityökalut käyttävät tietokantaliitynnän funktioita liittyäkseen konfiguraatio-tietokantaan. Tietokantaliityntä ja tietokanta integroivat ja yhdenmukaistavat käytettäviä työkaluja. Päättelysäännöillä korvataan vaikea ja virheherkkä konfiguraation vaihe, joka on tehty ennen manuaalisesti. Verkkokonfiguraattorin tulisi parantaa suunnittelu- ja konfigurointiproseduurin suoritusnopeutta ja vähentää siinä mahdollisesti tehtäviä virheitä.

Hajautusalustana käytettiin LON-kenttäväylää, joka soveltuu hyvin toiminnalliseen hajautukseen. Verkkokonfiguraattori toimi hyvin LON-kenttäväylän laitteiden automaattisena konfiguraattorina. Prototyyppi suoritti konfiguroinnit oikein, mutta suorituskyky ei ratkaisulla ollut kovin hyvä. Suorituskyvyn heikkous johtui pääasiassa tietokantaliitynnän hitaudesta ja koodin optimoimattomuudesta. Työn tuloksena voidaan todeta, että konfigurointiproseduuria voidaan automatisoida ja näin vähentää kenttäväyläverkon suunnittelussa ja konfiguroinnissa tehtävää työtä.

Jatkossa voitaisiin mennä automatisoinnissa pitemmälle ja tutkia plug-in-konfiguroinnin mahdollisuuksia. Plug-in-konfigurointiratkaisu mahdollistaisi yhä automaattisemman konfigurointiproseduurin.

LÄHTEET

1. Coulouris, G., Dollimore, J. & Kindberg T. 1994. Distributed Systems. Addison-Wesley Publishing Company, Wokingham. 644 s.

2. Vähämäki, O. 1997. LON-väylän käyttö sähkönjakeluverkon suojaus- ja ohjausjärjestelmissä. ABB Transmit Oy. 10 s.

3. Sirkkala, H. 1995. Kahden kenttäväylän välinen yhdyskäytävä. Diplomityö. Oulun yliopisto, sähkötekniikan osasto, Oulu. 74 s.

4. Madan, P. LonWorks technology for intelligent distributed interoperable control networks. Echelon Corporation, Palo Alto. 11 s.

5. Stallings, W. 1985. Data and computer communications. Macmillan Publishing Company, New York. 594 s.

6. Pietarinen, P., Honkanen, T. & Hyvärinen, J. 1997. Älykkyyden hajauttaminen LVIS-järjestelmien automaatiossa. HAJAÄLY-projektin loppuraportti. Espoo: Valtion teknillinen tutkimuskeskus. 106 s. + liitt. 6 s. (VTT Tiedotteita 1844.)

7. Kukkonen, A. 1997. Vapaaseen kilpailuun yhteisellä standardilla. Automaatioväylä, No. 7, s. 8–15.

8. Hänninen, V. 1996. Liitäntä kenttäväylään! Prosessori, No. 8, s. 35–37.

9. Korhonen, V. 1996. Beginners guide to LonWorks. ABB Relays, Vaasa. 30 s.

10. Korja, H. 1995. Erään kansainvälisen kenttäväylästandardin toiminta-analyysi käyttäjän näkökulmasta. Diplomityö. Oulun yliopisto, sähkötekniikan osasto, Oulu. 198 s.

11. LonWorks Product Line Brief 1990. Motorola. 19 s.

12. The SNVT Master List and Programmer’s Guide 1996. Echelon.

13. The SCPT Master List 1995. Echelon.

14. Projektiraportti IEC 1131-3: Työkalujen käyttökokemuksia 1996. VTT. 53 s.

15. Lewis, R. W. 1995. Programming industrial control systems using IEC 1131-3. The Institution of Electrical Engineers, London. 293 s.

16. Oueichek, I. 1996. Dynamic Configuration Management in the Guide Object-Oriented Distributed System. In: 3rd International Configurable Distributed Systems, May 6–8, Annapolis, USA.

17. Lon Network Tool, User's manual 1997. Software Engineers Oy.

18. Lon Network Agent, User's manual 1997. Software Engineers Oy.

19. Heikkinen, J. 1997 Spesification of network configurator for LON-bus based substation automation system. VTT Elektroniikka. 20 s.

20. http://www.profibus.com

21. Alanen J.& Virtanen, A. 1994. Ylemmän kerroksen CAN-kommunikointiarkkitehtuurit. Espoo: Valtion teknillinen tutkimuskeskus. 61 s. + liitt. 2 s. (VTT Tiedotteita 1561.)

22. Darscht, P., Frigeri, A. H.& Pereira, C. E. 1995. Building up object-oriented industrial automation systems: experiences interfacing active objects with technical plants. In: IEEE International Workshop on Factory Communications Systems, October 4–6, Leysin, Switzerland.

23. http://www.isa-online.org/journals/intech/

24. http://hallen.ele.kth.se/~willi/Fieldbus.html

25. Berghoff, J., Drobnik, O., Lingnau, A. & Mönch, C. 1996. Agent-based configuration management of distributed applications. In: 3rd International Configurable Distributed Systems, May 6–8, Annapolis, USA.

26. Hänninen, V. 1994. Kenttälaitteet omaan väylään. Prosessori, No. 1, s. 39–41.

27. Kataja, K., Rasimus, T. & Metsikkö, A. 1985. Ohjelmoitava logiikka. Suomen Sähköurakoitsijaliitto ry., Helsinki. 142 s.

28. LonWorks, yleinen tekninen kuvaus 1993. Avnet Nortec. 12 s.

29. Siirtola, J. & Alanen, J. 1991. Kenttä- ja ajoneuvoväylät. Metalliteollisuuden kustannus, Helsinki. 44 s.

30. http://www.plcs.net

31. http://vigna.cimsi.cim.ch/tai/BDC/in/BDC4_7.html

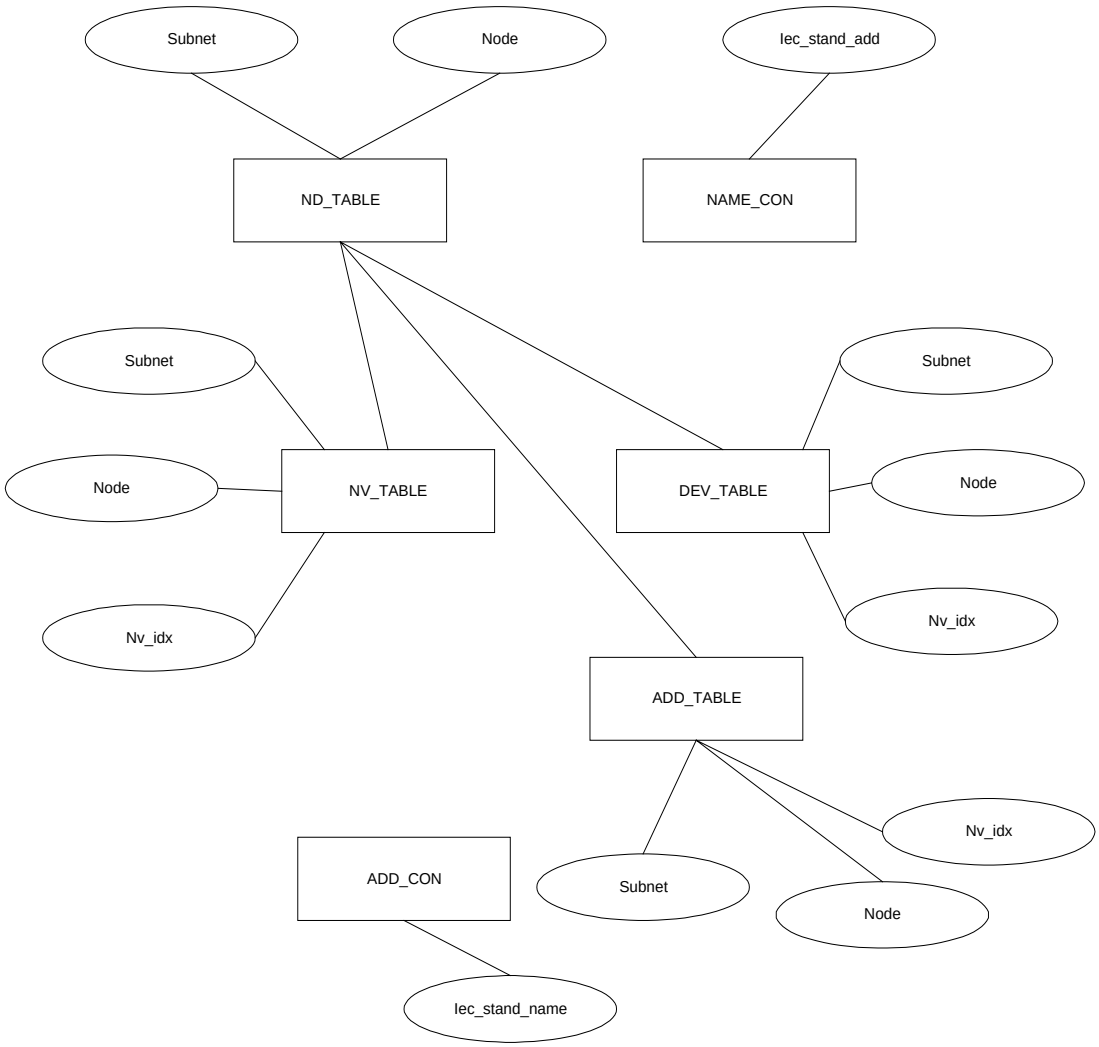
32. Wooldridge, M. J. & Jennings, N. R. 1994. Intelligent Agents. ECAI-94 Workshop on Agent Theories, Architecture, and Languages, August, Amsterdam, Netherlands. 407 s.

33. Solid Server Technical Description 1996. Solid IT Ltd. 58 s.

34. Cattell, R. G. G. 1997. The Object Database Standard: ODMG 2.0. Morgan Kaufman Publishers, Inc., San Francisco. 270 s.

35. http://www.solidtech.com/support/pg/prguide.htm

Tietokannan ER-malli LIITE 1



Tietokannan attribuuttien lyhenteet ja tietotyypit LIITE 2

Attribuutin nimi	Lyhenne	Tietotyyppi (c,SQL)
Subnet	SUBNET	int, SMALLINT
Node	NODE	int, SMALLINT
Device type	DEV_TYPE	unsigned char, VARCHAR
Extended address	EXT_ADD	unsigend char, VARCHAR
Address entries	ADD_ENT	unsigned char, VARCHAR
Location	LOCATION	unsigned char, VARCHAR
Priority	PRIO	int, TINYINT
Network variable name	NV_NAME	unsigned char, VARCHAR
Authentication	AUTH	int, TINYINT
Repeat timer	REP_TMR	int, TINYINT
Retry	RETRY	int, TINYINT
Receive timer	REC_TMR	int, TINYINT
Transaction timer	TRNS_TMR	int, TINYINT
Network variable index	NV_IDX	int, TINYINT
Standard network variable type	SNVT	int, TINYINT
Direction	DIR	int, TINYINT
Data length	DATALEN	int, TINYINT
Address index	ADD_IDX	int, TINYINT
Turnaround	TROUND	int, TINYINT
Connection name	CON_NAME	unsigned char, VARCHAR
Selector	SEL	int, SMALLINT
Address type	ADD_TYPE	int, TINYINT
Service	SER	int, TINYINT
Entry data	ENTRY_DATA	double, DOUBLE
IEC standard address	IEC_STAND_ADD	unsigned char, VARCHAR
IEC standard name	IEC_STAND_NAME	unsigned char, VARCHAR
Project	PROJECT	unsigned char, VARCHAR

Address Tablen ja NVCT:n rakenne LIITE 3

Network variable configuration table

1st byte:

priority	(1 bit)
direction	(1 bit)
selector_high	(6 bits)

2nd byte:

selector_low	(8 bits)

3rd byte:

turnaround	(1 bit)
service	(2 bits)
authentication	(1 bit)
address index	(4 bits)

Address table

1st byte:

address type	(8 bits)

2nd byte:

node id (0-127)	(8 bits)

3rd byte:

repeat timer	(4 bits)
retry	(4 bits)

4rd byte:

receive timer	(4 bits)
transaction timer	(4 bits)

5rd byte:

subnet id (0-255)	(8 bits)