LAPPEENRANNAN TEKNILLINEN KORKEAKOULU
Tietotekniikan osasto

OLIOSUUNTAUTUNUT ATM-KYTKIMEN OHJAUSOHJELMISTO

Diplomityön aihe on hyväksytty Lappeenrannan teknillisen korkeakoulun tietotekniikan osaston osastoneuvostossa 25.8.1999.

Työn ohjaaja ja tarkastaja: dosentti, FT Olli Martikainen

Lappeenrannassa 29.11.1999

Timo Pärnänen
Orkoniitynkatu 2 C 16
53850 Lappeenranta

TIIVISTELMÄ

Tekijä: Pärnänen, Timo Tapani

Tutkielman nimi: Oliosuuntautunut ATM-kytkimen ohjausohjelmisto

Osasto: Tietotekniikan osasto

Vuosi: 1999

Paikka: Lappeenranta

Diplomityö. Lappeenrannan teknillinen korkeakoulu.

60 sivua, 28 kuvaa, 7 taulukkoa ja 3 liitettä.

Tarkastaja dosentti, FT Olli Martikainen.

Avainsanat: ATM, CORBA, kytkin, ohjausohjelmisto, UML

Erilaisten tiedonsiirtotarpeiden lisääntyminen, perinteisten data- ja puhelinverkkojen siirtokapasiteetin rajoittuneisuus ja digitaalitekniikan kehittyminen ovat johtaneet laajakaistaisen digitaalisen monipalveluverkon määrittelyyn. Tämän verkkokonseptin toteuttajaksi kehitettiin ATM-tekniikka, joka ei ole kuitenkaan saavuttanut odotettua suosiota. Tietoliikenneverkkojen infrastuktuuri tullee säilymään jatkossakin heterogeenisena.

Tietoliikenneohjelmistojen tulisi sopeutua kulloisenkin verkkoinfrastruktuurin mukaisesti. Erilaisten olioteknologioiden kehittyminen mahdollistaa ohjelmistojen skaalautuvuuden ja nopean kehityksen. Olioteknologiat ovat vääjäämättömästi syrjäyttämässä perinteiset proseduraaliset ohjelmointikielet. Myös koko ohjelmistokehitysprosessiin on alettu kiinnittää entistä enemmän huomiota.

Tämä diplomityö kuuluu Teknillisen korkeakoulun tietoliikenneohjelmistojen ja multimedian laboration toteuttaman SCOMS-projektin tuloksiin. Työ on tehty osittain Toimialaverkot-projektissa. Siinä keskitytään ATM-kytkimen ohjausohjelmiston oliopohjaiseen toteuttamiseen. Työssä esitellään kytkimen ohjausohjelmiston vaatimuksia ATM-tekniikan näkökulmasta, toteutuksessa käytettyjä oliotekniikoita sekä syntyneen ohjausohjelmiston toteutusta ja rakennetta. Ohjelmisto on suunniteltu SCOMS-projektin tarpeisiin tukea erilaisten verkkotekniikoiden yhteistoimintaa monikäyttökytkimessä.

ABSTRACT

Author: Pärnänen, Timo Tapani

Title: Object-oriented controlling software of ATM switch

Department: Department of Computer Science

Year: 1999

Place: Lappeenranta

Master of Science Thesis. Lappeenranta University of Technology.

60 pages, 28 pictures, 7 tables and 3 appendices.

Supervisor docent, PhD Olli Martikainen

Keywords: ATM, CORBA, switch, controlling software, UML

Increase of the different data transfer needs, the limitation of the transfer capacity of the traditional data and telephone networks, and the development of the digital technique, have led to the definition of the broadband integrated services digital network. ATM technique was developed to realize this network outline, but it has not reached an expected popularity. The infrastructure of data communications networks will probably remain heterogeneous also in the future.

The data communication software should be able to adapt themselves according to different network infrastructures. The development of different object-oriented technologies makes scaleable software and quick development possible. The object-oriented technologies are displacing the traditional programming languages undeniably. Also entire software development process has been paid more attention than before.

This thesis is part of results of the SCOMS project carried out by the Telecommunication Software and Multimedia Laboratory at Helsinki University of Technology. The work has been done partly in the Industrial Networks project and it concentrates on the object-oriented implementation of the control software for ATM switch. This thesis shows the requirements of control software for switching hardware in the ATM point of view, used object-oriented technologies and implementation and structure of software which has been created. The software has been designed for the needs of the SCOMS project to support interworking of different network techniques in multidiscipline switch.

ALKUSANAT

Diplomityö on tehty Teknillisen korkeakoulun tietoliikenneohjelmistojen ja multimedian laboratorion toteuttamissa TOVE- ja SCOMS-projekteissa. Työ kuuluu osana SCOMS-projektin dokumentaatiota.

Kiitän Olli Martikaista työn ohjauksesta ja tarkastamisesta sekä hänen moniin visioihinsa perustuvista mielenkiintoisista ja opettavaisista tutkimus-projekteista.

”Pomppu”-tiimille kiitokset vapaa-ajan urheiluharrastuksista, jotka antoivat voimaa tämän työn kirjoittamisessa.

Kiitokset myös kotiväelle opiskelun tärkeyden painottamisesta ja opintojen tukemisesta, valmistuihan se vihdoinkin, vaikka välillä taisi usko jo loppua. Loistavaa urheilutoimittajaa lainatakseni: ”Se on siinä”.

SISÄLLYSLUETTELO

LYHENTEET 3
1 JOHDANTO 7
2 ATM 9
2.1 ATM-verkot 9
2.2 Yhteydet ATM-verkoissa 10
2.3 Merkinanto ja yhteydenmuodostus 12
3 REITITYS 15
3.1 Osoitteet 15
3.2 Reititysprotokolla 16
4 VERKONHALLINTA 19
4.1 SNMP ja MIB 19
4.2 ILMI 20
5 KYTKENTÄKENTÄN OHJAUS 22
5.1 GSMP 22
5.1.1 Johdanto 23
5.1.2 Rakenne 23
5.1.3 Pakettien kapselointi 23
5.1.4 Sanomat 24
5.2 FSR-API 28
6 OLIOMENETELMÄT 30
6.1 UML 30
6.2 Suunnittelumallit 33
6.3 CORBA 35
6.3.1 Arkkitehtuuri 35
6.3.2 ORB 36
6.3.3 Rajapinnan kuvauskieli 37
6.4 OVOPS++ 38
6.4.1 Johdanto 38
6.4.2 Rakennuselementit 39

6.4.3 Sanomat 40
6.4.4 Ajanjako 40
7 OHJAUSOHJELMISTON TOTEUTUS 42
7.1 TOVE- ja SCOMS-projektit 42
7.2 Hajautettu ohjelmisto 43
7.2.1 ILMI- ja SNMP-agentit 45
7.3 Kytkinmoduuli 45
7.3.1 Kytkinolio 45
7.3.2 Portit ja linkit 47
7.3.3 Kytkentäkentän ohjaus 49
7.4 Konfigurointi 49
7.5 Puhelunmuodostus 50
8 JOHTOPÄÄTÖKSET 53

LÄHTEET
LIITE 1. Luokkakaavio porttien ja linkkien rakenteesta.
LIITE 2. Porttien ja linkkien alustus GSMP:llä.
LIITE 3. SCOMS-kytkimen konfiguraatiotiedosto.

LYHENTEET

AAL	ATM Adaptation Layer
AFI	Authority and Format Identifier
ANSI	American National Standards Institute
API	Application Programming Interface
ASN.1	Abstract Syntax Notation One
ATM	Asynchronous Transfer Mode
BCD	Binary Coded Decimal
B-ICI	B-ISDN Inter-Carrier Interface
B-ISDN	Broadband Integrated Services Digital Network
B-ISUP	B-ISDN User Part
CAC	Call/Connection Admission Control
CLP	Cell Loss Priority
CORBA	Common Object Request Broker Architecture
CPCS	Common Part Convergence Sublayer
CVOPS	C-based Virtual Operating System
DCC	Data Country Code
DCF	Distributed Component Framework
DSP	Domain Specific Part
DSS2	Digital Signaling Subscriber No. 2
DTL	Designated Transit List
ESI	End System Identifier
FCF	Fabric Control Function
FSR	Frame Synchronized Ring
GCAC	Generic Call Admission Control
GFC	Generic Flow Control
GIOP	General Inter-ORB Protocol
GSMP	General Switch Management Protocol

GUI	Graphical User Interface
HEC	Header Error Check
HO-DSP	Higher Order Domain Specific Part
ICD	International Code Designator
IDI	Initial Domain Identifier
IDL	Interface Definition Language
IDP	Initial Domain Part
JVOPS	Java Virtual Operating System
IETF	Internet Engineering Task Force
IIOP	Internet Inter-ORB Protocol
IISP	Interim Inter-Switch Signaling Protocol
ILMI	Integrated Local Management Interface
	(Interim Local Management Interface)
IME	Interface Management Entity
INGW	Intelligent Network Gateway
IP	Internet Protocol
ISDN	Integrated Services Digital Network
ISO	International Standardisation Organisation
ISUP	ISDN User Part
ITU-T	International Telecommunication Union –
	Telecommunication standardization sector
LAPD	Link Access Procedures on the D-channel
LGN	Logical Group Node
LLC	Logical Link Control
LTKK	Lappeenrannan teknillinen korkeakoulu
MAC	Media Access Control
MIB	Management Information Base
MTP	Message Transfer Part
NNI	Network Node Interface
	Network Network Interface
NSAP	Network Service Access Point

OMA	Object Management Architecture
OMG	Object Management Group
OMT	Object Modeling Technique
OOSE	Object-Oriented Software Engineering
ORB	Object Request Broker
OUI	Organizationally Unique Identifier
OVOPS	Object-oriented Virtual Operating System
PDH	Plesiochronous Digital Hierarchy
PDU	Protocol Data Unit
PGL	Peer Group Leader
PID	Protocol Identifier
PNNI	Private Network Network Interface
PT	Payload Type
PVC	Permanent Virtual Connection
QoS	Quality of Service
RFC	Request For Comments
SAAL	Signaling ATM Adaptation Layer
SAR	Segmentation And Reassembly
SCCP	Signaling Connection Control Part
SCOMS	Software Configurable Multidiscipline Switch
SDH	Synchronous Digital Hierarchy
SEL	Selector
SMI	Structure of Management Information
SNAP	SubNetwork Attachment Point
SNMP	Simple Network Management Protocol
SSCF	Service Specific Coordinate Function
SSCOP	Service Specific Coordination Oriented Protocol
SS7	Signaling System No. 7
STM	Synchronous Transport Module
SVC	Switched Virtual Connection
TAXI	Transparent Asynchronous Tranceiver Interface

TCAP	Transaction Capabilities Application Part
TCP	Transmission Control Protocol
TEKES	Teknologian kehittämiskeskus
TKK	Teknillinen korkeakoulu
TOVE	Toimialaverkot
UDP	User Datagram Protocol
UML	Unified Modeling Language
UNI	User Network Interface
VCI	Virtual Channel Identifier
VPI	Virtual Path Identifier
VTT	Valtion teknillinen tutkimuskeskus
XML	Extensible Markup Language

1 JOHDANTO

Tietokoneiden ja oheislaitteiden suorituskyvyn huima kasvu mahdollistavat yhä monipuolisempien palvelujen toteuttamisen. Kiihtyvällä vauhdilla kasvava verkottuminen ja käyttäjien kiinnostus erilaisiin verkkopalveluihin mahdollistavat laajamittaisen liiketoiminnan sekä palveluntarjoajille että verkko-operaattoreille. Perinteisten televerkkojen tarjoamien puhelin-palvelujen rinnalle kehitellään yhä monipuolisempia multimediapalveluja.

Televerkkojen rinnalle on rakennettu erilaisiin tiedonsiirtotarpeisiin soveltuvia verkkoja. Kustannussyistä usean rinnakkaisen verkon rakenta-minen ei ole pidemmällä tähtäimellä kannattavaa. Kehitys on kulkenut kohti digitaalista laajakaistaista monipalveluverkkoa (B-ISDN, Broadband Integrated Services Digital Network), jonka toteuttamiseen kehitettiin ATM-tekniikka (Asynchronous Transfer Mode). Se tukee palvelunlaatu-määrityksiä ja tarjoaa joustavasti eri käyttäjille räätälöityjä tiedonsiirto-yhteyksiä.

ATM-tekniikka ei ole kuitenkaan lyönyt itseään läpi odotetulla tavalla. Se näyttää jäävän runkoverkkojen tekniikaksi, kapeakaistaisen ISDN:n tarjotessa yhteydet loppukäyttäjille ja TCP/IP:n (Transmission Control Protocol/Internet Protocol) hallitessa lähiverkkoja. ATM-kytkinten rinnalle ollaan kehittämässä monikäyttökytkimiä, jotka mahdollistavat erilaisten tiedonsiirtoverkkojen yhdistämisen samaan verkon solmuun.

Verkon solmun ohjausohjelmiston on oltava helposti laajennettava kulloisenkin verkkokonseptin mukaisesti. Olioteknologioiden kehittyminen mahdollistaa modulaarisen ja skaalautuvan ohjelmiston toteuttamisen. Suunnittelumallit, UML (Unified Modeling Language), CORBA (Common Object Request Broker Architecture) ja sovelluskehykset tarjoavat puitteet, joilla nämä laajennettavuusvaatimukset voidaan täyttää.

Diplomityön tavoitteena on kytkimen ohjausohjelmiston rungon toteuttaminen oliotekniikoita hyödyntäen. Tähän liittyy porttien, linkkien ja kytkentäkentän ohjauksen sekä ohjausohjelmiston muiden osien yhteistoiminnan toteuttaminen. Ohjausohjelmisto tukee aluksi vain ATM-yhteyksiä, mutta on helposti laajennettavissa tukemaan myös kapeakaistaisia yhteyksiä.

Teoriaosassa on aluksi lyhyt johdanto ATM-tekniikkaan. Seuraavaksi esitellään reitityksen, verkonhallinnan ja kytkentäkentän ohjauksen näkökohtia ATM-yhteyksien kannalta. Oliomenetelmien tutkiminen ja soveltaminen ovat tärkeä osa työtä, joten oliotekniikoiden esittely on varsin laaja osa työtä. Käytännönosuus kuvaa kytkimen ohjausohjelmiston toteutusta ja syntyneen ohjelmiston rakennetta.

2 ATM

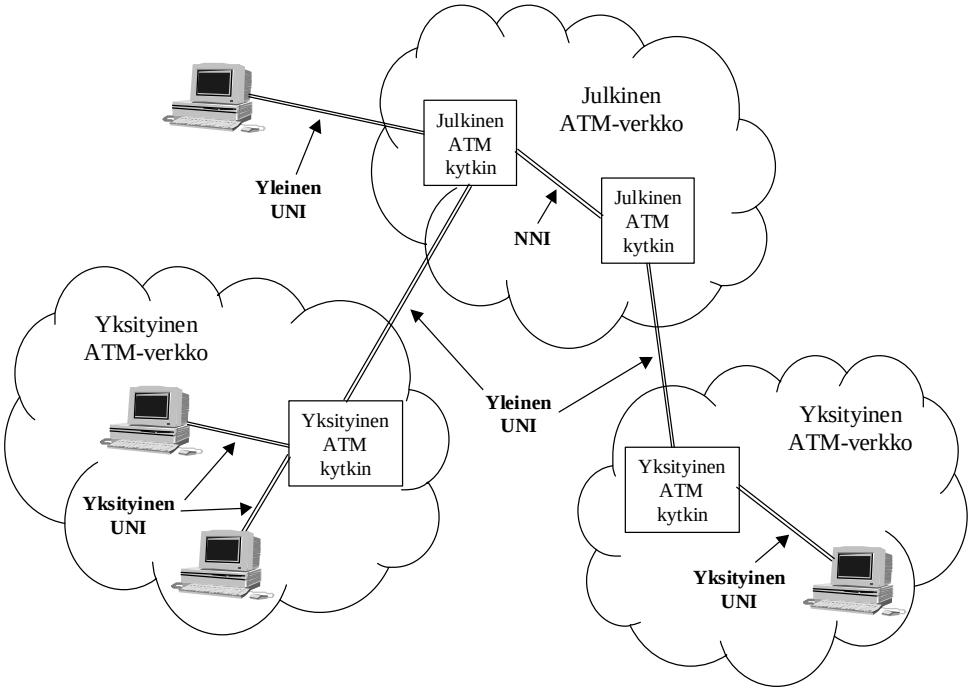
Olemassa olevat puhelin-, data- ja tietokoneverkot on alunperin suunniteltu vain yhtä käyttötarkoitusta varten. Pidemmällä tähtäimellä tällainen yhden-asianverkkojen rakentaminen ei ole kustannussyistä järkevää. Tästä syystä on suunniteltu tietoverkkoa, joka pystyy tukemaan keskenään aivan eri tyyppisiä tiedonsiirtotarpeita. Kehitys on edennyt kohti laajakaistaista digitaalista monipalveluverkkoa (B-ISDN). B-ISDN on pelkkä konsepti, jonka toteuttamiseksi tarvitaan joukko erilaisia tekniikoita. Tämän konseptin toteutusten merkittävimpänä piirteenä on ATM-tekniikan käyttö siirto-protokollana /1/.

ATM syntyi ja kehittyi pääasiassa suurten teleoperaattorien muodostamassa standardointijärjestössä ITU-T:ssä (International Telecommunication Union–Telecommunication standardization sector). ITU-T keskittyy ennen kaikkea julkisiin verkkoihin. B-ISDN-verkkokonsepti on nimenomaan ITU-T:n määrittelemä. ITU-T on standardointiorganisaationa suuri ja siksi sen standardointiprosessi on verraten hidas. ITU-T:n standardit ovat usein hyvin laajoja ja ne jättävät paljon avoimia piirteitä. Nopeuttaakseen standardointia, tietokonealalla toimivat valmistajat muodostivat yhteenliittymän, ATM Forumin. Sen tarkoituksena on valmistaa standardien kaltaisia toteutus-sopimuksia, joita eri valmistajat noudattavat. Toteutussopimukset määrit-televät mitä standardeja käytetään ja miten. /1/.

2.1 ATM-verkot

ATM-verkko koostuu fyysisillä linkeillä toisiinsa kytketyistä päätelaitteista ja verkon solmuista. Solmut ovat joko kytkimiä tai keskittimiä. Kytkimen tehtävänä on suorittaa yhteyksien ristiinkytkentää. Keskittimet kokoavat (multipleksoivat) useita yhteyksiä yhdeksi suuremmaksi tietovirraksi ja päinvastoin. /1/

ATM-verkot jaetaan yksityisiin ja julkisiin verkkoihin. Suurimmat erot näiden verkkotyyppien välillä on verkon- ja liikenteenhallinnassa. Verkko-liittymät vaihtelevat riippuen siitä, liitytäänkö julkiseen vai yksityiseen verkkoon. Myös verkkojen välinen liittymä on erilainen riippuen toisiinsa liitettävien verkkojen tyypistä. ATM Forum määrittelee käyttäjän ja verkon välisen rajapinnan (UNI, User Network Interface) sekä verkkojen ja verkon solmujen välisen rajapinnan (NNI, Network Node Interface). Erot ovat lähinnä merkinannossa, NNI-rajapinnan on tuettava myös reititystä. /1/



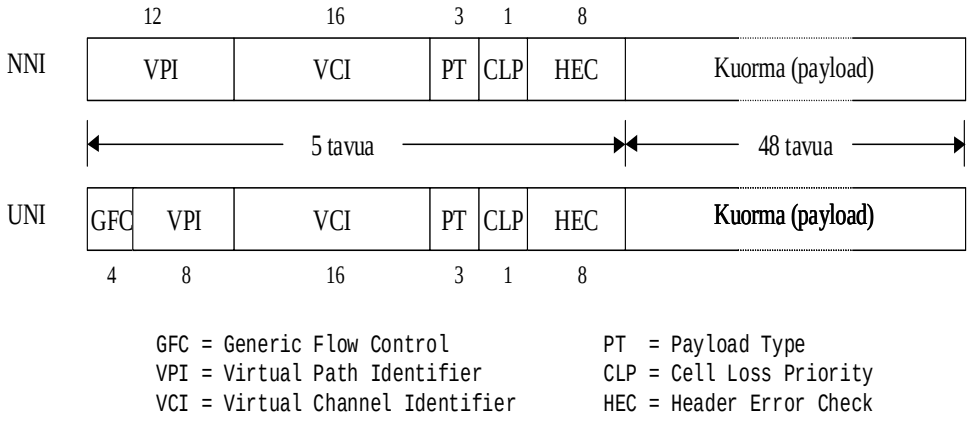
Kuva 1. ATM-verkon rakenne.

2.2 Yhteydet ATM-verkoissa

ATM on yhteydellinen tekniikka, ja siinä on ominaisuuksia sekä piirikytkentäisestä että pakettikytkentäisestä tiedonsiirrosta. ATM-tiedon-siirron paketit eli solut kytketään lähteeltä kohteelle. Paketin pienuuden ansiosta kytkentä on nopeaa, joten ohjauksen voidaan ajatella toimivan

piirikytkentäisen verkon tapaan. ATM:n tiedonsiirtoyhteyttä kutsutaankin myös virtuaaliseksi piiriksi (virtual circuit). /1/

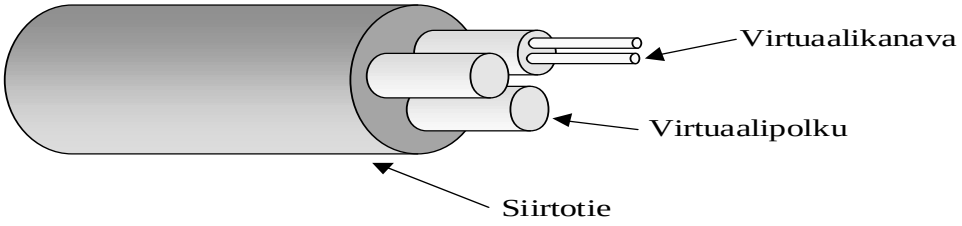
ATM:n protokollatietoyksikköä (PDU, Protocol Data Unit) kutsutaan soluksi. Solun pieni koko mahdollistaa nopean pakettikytkennän, mikä on välttämätöntä viiveherkille yhteyksille. ATM-solu on vakiomittainen, kooltaan 53 tavua. Solu koostuu viiden tavun ohjaustiedosta eli otsakkeesta (header) ja 48:n tavun kuormasta (payload). Solun koko on kompromissi datan- ja puheensiirtotarpeista. Solun otsakkeen rakenne on erilainen UNI- ja NNI-rajapinnoissa. Otsakkeen kenttiä ovat: yleinen vuonohjauskenttä (GFC, Generic Flow Control), virtuaaliyhteyden tunnisteet (VPI/VCI, Virtual Path Identifier/Virtual Channel Identifier), kuorman tunniste (PT, Payload Type), hävikkiprioriteetti (CLP, Cell Loss Priority) ja otsakkeen tarkistussumma (HEC, Header Error Check). /1//2/



Kuva 2. ATM-solun rakenne UNI- ja NNI-rajapinnoissa.

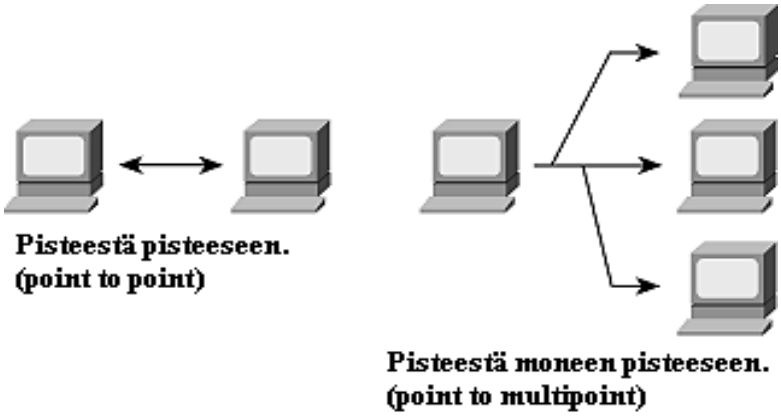
ATM- eli virtuaaliyhteydet muodostavat kaksitasoisen hierarkian siirtotielle. Alemman tason virtuaaliyhteyksiä kutsutaan virtuaalikanaviksi (virtual channel), joita varsinaisesti käytetään tiedonsiirtoon. Ylemmän hierarkiatason muodostavat virtuaalipolut (virtual path), jotka sisältävät joukon virtuaalikanavia. Virtuaalipolkuyhteyksiä käytetään resurssien-hallinnan helpottamiseksi. Samoja ominaisuuksia (esim. viiveherkät

yhteydet) sisältävät virtuaalikanavat voidaan niputtaa yhdeksi virtuaalipoluksi ja käsitellä samalla tavalla. /1/ /3/



Kuva 3. Siirtotie ja virtuaaliyhteydet.

ATM tarjoaa kahdentyyppisiä yhteyksiä: pisteestä pisteeseen (point-to-point) ja pisteestä moneen pisteeseen (point-to-multipoint). Ensimmäinen näistä on normaali kahden käyttäjän välinen yhteys. Jälkimmäisen avulla tarjotaan ryhmäjakelu (multicast). Yleisjakelu (broadcast) on mahdollista ryhmäjakeluna, jossa kaikille käyttäjille muodostetaan yhteys. /3/ /1/



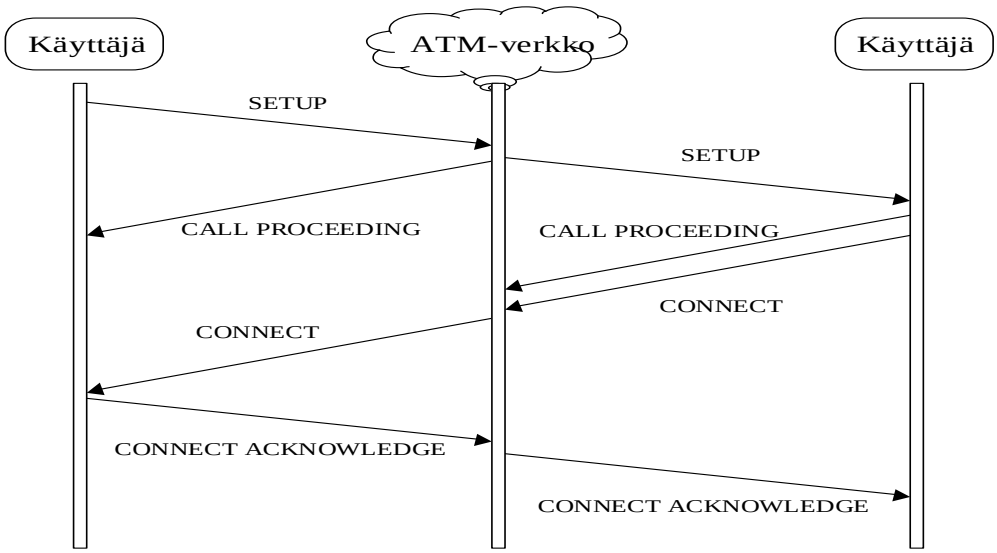
Kuva 4. Yhteystyypit

2.3 Merkinanto ja yhteydenmuodostus

ATM:ssä yhteydet voivat olla joko kiinteitä (PVC, Permanent Virtual Connection) tai kytkentäisiä (SVC, Switched Virtual Connection). Kiinteät yhteydet varataan hallintaosan toiminnoilla ja ne muodostetaan automaattisesti uudelleen verkon uudelleenkäynnistyksen yhteydessä.

Kytkentäiset yhteydet muodostetaan ja puretaan dynaamisesti merkinannon avulla. Ne eivät säily esim. verkkoviasta johtuvan uudelleenkäynnistyksen yhteydessä.

Käyttäjien ATM-päätelaitteiden väliset yhteydet (puhelut) muodostetaan merkinannon avulla. Yhteydenmuodostukseen käytetään erillistä merkinantokanavaa (VPI=0, VCI=5). Merkinanto tarjoaa puhelunohjausproseduurille kolme funktiota: puhelunmuodostus, puhelun purku ja puhelun tilan seuranta (status). ATM:ssä merkinantopino koostuu fyysisen ja ATM-kerroksen päälle määritellystä erityisestä merkinantosovituskerroksesta (SAAL, Signaling ATM Adaptation Layer) sekä UNI- tai NNI-merkinantoprotokollasta. SAAL tarjoaa merkinantoinformaation luotettavan siirron. /1//2/



Kuva 5. Yhteydenmuodostusmenettely UNI-rajapinnassa.

ATM Forumin UNI on nyt versiossa 4.0 ja se perustuu ITU-T:n Q.2931-standardiin (DSS2, Digital Signaling Subscriber No. 2). Verkkosignalointiin ITU-T määrittelee B-ISUP:n (B-ISDN User Part) ja ATM Forum yksityisiin verkkoihin PNNI:n (Private Network Network Interface) sekä julkisiin verkkoihin B-ICI:n (B-ISDN Inter-Carrier Interface).

ATM-verkkojen etuna on, että muodostettavat yhteydet voidaan räätälöidä asiakkaan tarpeiden mukaan. Kullekin virtuaaliyhteydelle määrätään liikenneluokka liikennekuvauksen perusteella. Liikenneluokka on osa liikennöintisopimusta. Liikennekuvaus koostuu laatu- (QoS, Quality of Service) ja liikenneparametreistä. Liikenneparametreillä kuvataan siirto-kapasiteetin tarvetta ja laatuparametreillä puolestaan minkälaista palvelua asiakas verkolta haluaa. Laatuparametreihin kuuluvat mm. kuormavirhe-suhde, soluhävikki sekä siirtoviive ja sen vaihtelun rajat. Liikenne-parametreillä määritellään kaistanleveys ja purskeiden koko. /1/

Yhteyden muodostukseen kuuluu resurssien varaus. Pääsynvalvonnan (CAC, Call/Connection Admission Control) tehtävänä on selvittää riittävätkö verkon resurssit muodostettavan yhteyden liikenne- ja laatuparametrien toteuttamiseen. Pääsynvalvonnalla on tärkeä rooli myös vuonohjauksessa ja ruuhkanhallinnassa. /1/

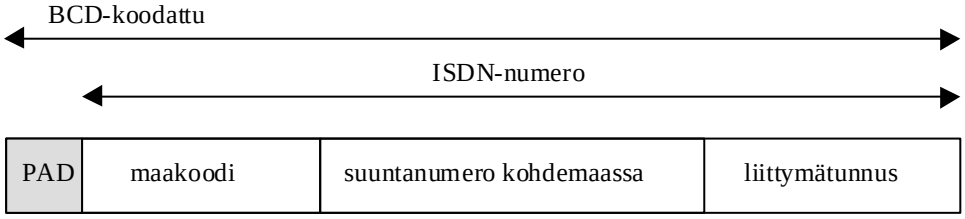
3 REITITYS

Koska ATM on yhteydellinen tekniikka, kuuluu yhteydenmuodostukseen reitin löytäminen ATM-päätepisteiden välille. Reititys tehdään ATM-osoitteiden perusteella. /1/

3.1 Osoitteet

ATM-verkoissa käytetään kahdentyyppisiä osoitteita. ITU-T:n määrittelemää E.164-standardin /4/ mukaista osoiteformaattia käytetään julkisissa ATM-verkoissa ja ATM Forumin määrittelemää NSAP-osoitetta (Network Service Access Point) yksityisissä ATM-verkoissa. Riippumatta käytettävästä osoitteistuksesta ATM-osoite on aina 20 oktetin mittainen.

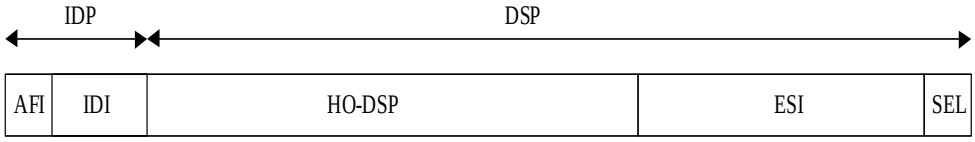
E.164-osoitteistus on tuttu puhelinnumeroformaatti. Se on alunperin tarkoitettu ISDN-verkkoihin. E.164-osoite on hierarkinen. Se muodostuu maakoodista, verkkoryhmänumerosta eli suuntanumerosta kohdemaassa ja liittymätunnuksesta.



Kuva 6. E.164-osoiteformaatti.

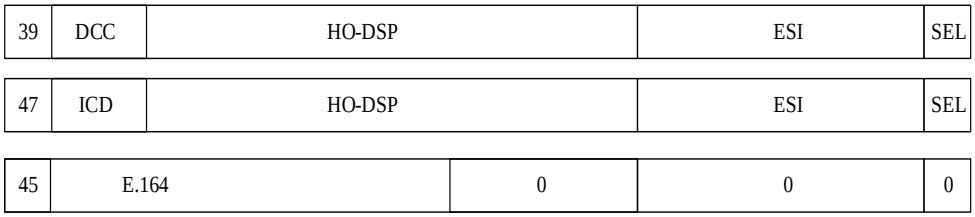
NSAP-osoite muodostuu kahdesta osasta. IDP (Initial Domain Part) eli taho kertoo osoitteen määrittelijän ja DSP (Domain Specific Part) osoitetyypin. IDP koostuu kahdesta kentästä: AFI (Authority and Format Identifier) kertoo osoitteiden virallisen määrittelijän ja IDI (Initial Domain Identifier) vastaavan tunnisteen. DSP sisältää tahon spesifisen osuuden (HO-DSP,

Higher Order Domain Specific Part), päätepisteen tunnisteen (ESI, End System Identifier) ja valitsinkentän (SEL, Selector).



Kuva 7. NSAP-osoitteen rakenne.

NSAP-osoitteelle on määritelty kolme koodaustapaa: DCC (Data Country Code), ICD (International Code Designator) ja E.164. DCC määrittelee ISO:n (International Standardisation Organisation) standardin mukaisen maakoodin, joka kertoo osoitteen sijainnin. ICD määrittelee Britannian standardointi-instituutin listaaman osoiteformaatista vastaavan kansain-välisen organisaation. Myös BCD-koodattua (Binary Coded Decimal) E.164-numerointia voidaan käyttää NSAP-osoitteen sisällä.



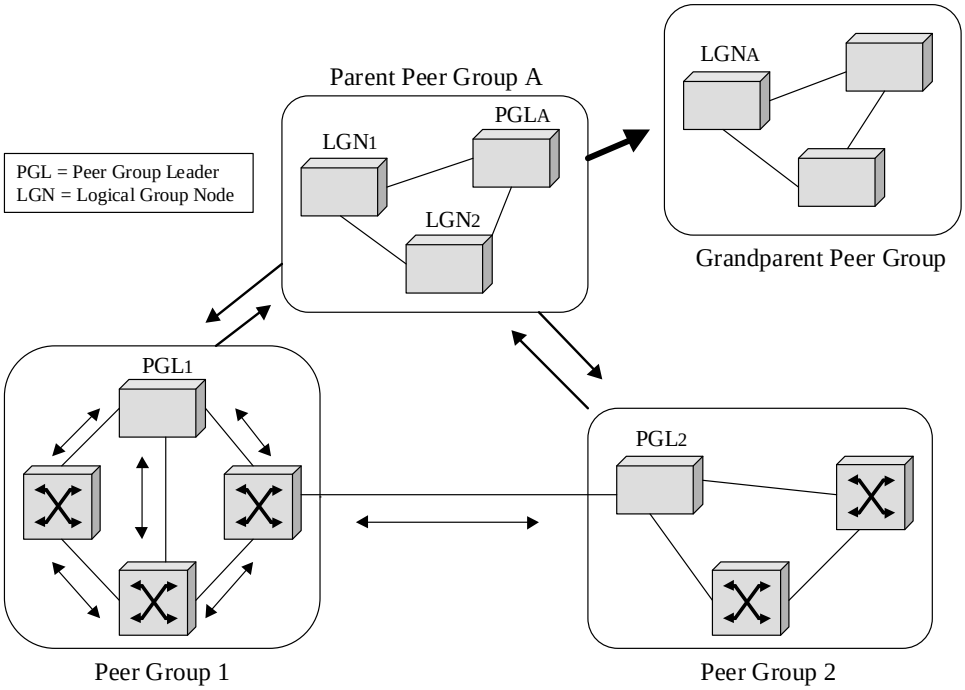
Kuva 8. NSAP-osoitteen koodausvaihtoehdot.

3.2 Reititysprotokolla

Aikaisemmin IISP (Interim Inter-Switch Signaling Protocol) oli ainoa standardi mekanismi ATM-reititykseen. Se perustui manuaalisesti konfiguroitaviin staattisiin reitteihin. IISP oli hyvin yksinkertainen väliaikaisratkaisu, kunnes ATM Forum julkisti vuonna 1996 PNNI-protokollan (Private NNI) /5/. Se oli ensimmäinen todellinen ATM-reititykseen ja kytkinten väliseen merkinantoon määritelty protokolla, joka tukee myös palvelunlaatuvaatimuksia (QoS) /6/.

Verkon solmut ylläpitävät tietoa verkon eri osien saavutettavuudesta ja tarjolla olevista resursseista. PNNI sallii ATM-kytkimen valita koko reitin verkon läpi kohteeseen asti. Verkon solmut arvioivat toisten solmujen kykyä taata laatuvaatimukset käyttäen GCAC-algoritmia (Generic Call Admission Control). Tuloksena saatu reitti talletetaan DTL-informaatioelementtiin (Designated Transit List), jota kuljetetaan yhteyspyynnön edetessä solmulta toiselle. Jos jokin solmu ei pystykään takaamaan pyydettyjä laatu-vaatimuksia, palautetaan yhteydenmuodostus takaisin lähimmälle solmulle (crankback), joka etsii uuden reitin kohteeseen /7/.

PNNI-protokolla käyttää hierarkista verkkorakennetta. Hierarkisuus mahdollistaa PNNI-reitityksen soveltamisen pienistä yksityisistä verkoista aina maailmanlaajuisiin ATM-verkkoihin asti.



Kuva 9. PNNI-reitityksen hierarkinen verkkomalli.

Verkon solmuista muodostetaan loogisia ryhmiä (peer group), joiden sisällä verkon laitteet tuntevat koko ryhmän topologian. Loogiset ryhmät voidaan koota edelleen ylemmän tason ryhmiksi (parent group). Topologiatietojen vaihto tehdään aina saman hierarkiatason loogisten ryhmien kesken /3/ /7/.

4 VERKONHALLINTA

Verkonhallintajärjestelmiä tarvitaan sekä lähi- että runkoverkkojen ylläpitoon. Useimmiten ne perustuvat hallinta-agenttimalliin. Yleisimmin käytetty verkonhallintaprotokolla on SNMP (Simple Network Management Protocol) /1/.

Seuraavassa esitellään lyhyesti SNMP-verkonhallintaprotokolla ja standardi MIB-rakenne (Management Information Base) hallittavan tiedon organisointiin. Lisäksi kuvataan ATM Forum:n määrittelemä paikallinen hallintarajapinta eli ILMI (Integrated Local Management Interface), joka määrittelee kuinka SNMP-protokollaa ja ATM-rajapinta-MIB:iä voidaan käyttää ATM-laitteiden hallintaan.

4.1 SNMP ja MIB

SNMP on suunniteltu yksinkertaiseksi ja helposti toteutettavaksi verkonhallintatyökaluksi TCP/IP-pohjaisiin verkkoihin. Termi SNMP viittaa itse asiassa joukkoon verkonhallintastandardeja, jotka määrittelevät itse verkonhallintaprotokollan (SNMP), hallintatietokannan (MIB) sekä joukon hallittavia olioita (SMI, Structure of Management Information) /8/.

SNMP-verkonhallinta-arkkitehtuurin osia ovat: hallinta-asema, hallinta-agentit, hallintainformaatio sekä verkonhallintaprotokolla. Verkonhallinta suoritetaan keskitetysti jostain työasemasta käsin kommunikoimalla kunkin verkkoelementin hallinta-agentin kanssa. Verkonhallinta perustuu hajautettuun tietokantamalliin, jossa hallittavat oliot on järjestetty standardin mukaiseksi puumaiseksi MIB-rakenteeksi. MIB on virtuaalinen tietovarasto, joka tarjoaa pääsyn hallittaviin ’olioihin’. Nämä oliot ovat itse asiassa hallinta-agentin muuttujia. Olioiden kuvaamiseen käytetään ASN.1-notaatiota (Abstract Syntax Notation One) /8/.

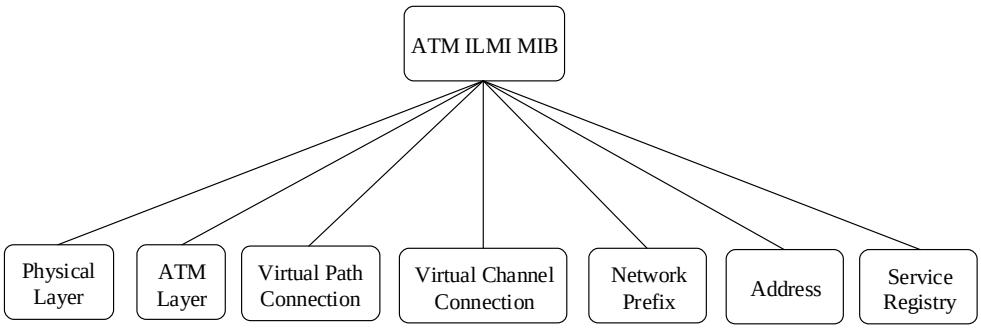
Koska SNMP:tä ajetaan yhteydettömän UDP-protokollan (User Datagram Protocol) päällä, myös SNMP on yhteydetön protokolla. SNMP sisältää neljä operaatiota informaation käsittelyyn /8/:

1. Get tietyn hallittavan tiedon kysyminen
2. Get-Next tiedon kysyminen MIB:iä selaamalla
3. Set hallittavan tiedon muuttaminen
4. Trap hallittava laite ilmoittaa jostain tapahtumasta

4.2 ILMI

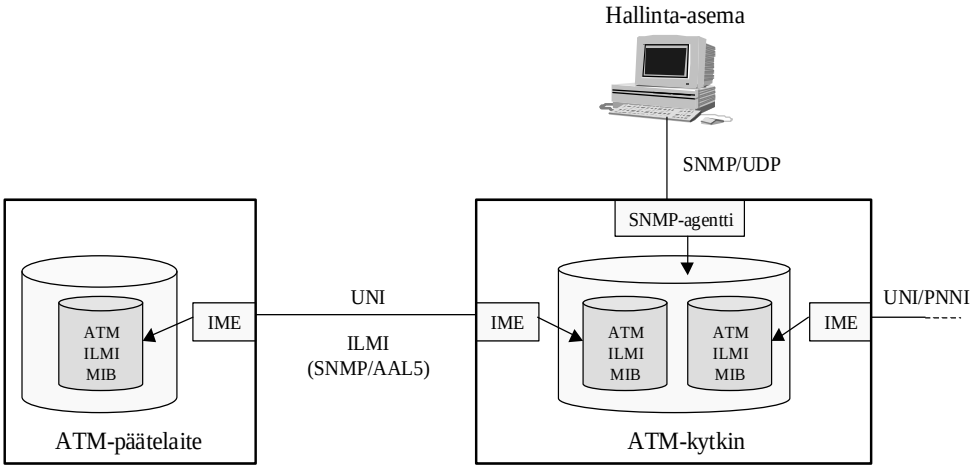
ATM Forum määritteli ILMI-standardin alunperin tilapäiseksi ratkaisuksi (Interim Local Management Protocol), koska ITU-T- ja ANSI-standardointi (American National Standards Institute) eivät edenneet odotetulla tavalla ATM-verkonhallinnan suhteen. Sittemmin ratkaisusta tuli pysyvä (Integrated Local Management Protocol). /9/

ILMI:n tehtävänä on tarjota ATM-rajapinnan fyysisen kerroksen ja ATM-kerroksen tila-, konfigurointi- ja ohjaustietoja liittyen virtuaalipolku- ja virtuaalikanavayhteyksiin. Jokainen ATM-laite sisältää yhden tai useamman ATM-rajapinnan (esim. UNI). Jokaista rajapintaa varten on olemassa joukko hallittavia olioita, jotka sisältävät ILMI-toiminnallisuuden tarvitsemat attribuutit. Nämä attribuutit on järjestetty standardiksi MIB-rakenteeksi (ILMI MIB). /9/



Kuva 10. ATM-ILMI-MIB:n puurakenne.

ATM-laitteessa on jokaista ATM-rajapintaa varten oma hallintaolio (IME, Interface Management Entity), joka tarjoaa ILMI-toiminnallisuuden ATM-rajapinnalle. Kun kaksi ATM-laitetta kytketään yhteen, ILMI-kommunikointi suoritetaan kahden IME-olion välillä. Kommunikointiprotokollaksi ILMI:iin on valittu SNMP. ILMI:ssä SNMP ei käytä UDP/IP-protokollaa vaan SNMP-sanomat pakataan AAL5-sanomiin. ILMI käyttää oletusarvoisesti VPI=0, VCI=16 virtuaalikanavaa. /9/



Kuva 11. ILMI-verkonhallinta.

ILMI tarjoaa UNI-rajapinnalle myös automaattisen osoitteenrekisteröinnin. Se mahdollistaa verkko-operaattoreiden valvoa paremmin osoitteiden varaamista ja vähentää oleellisesti virhemahdollisuutta, kun käyttäjälle epäystävällinen 20-oktetin mittainen ATM-osoite varataan automaattisesti. /6/ Osoitteenrekisteröinnissä ATM-kytkin lähettää päätelaitteelle listan mahdollisia osoitteiden verkkoetuliitteitä (network prefix). Päätelaite liittää näihin oman järjestelmäkohtaisen osuutensa (ESI) eli käytännössä MAC-osoitteensa (Media Access Control) ja palauttaa osoitteet takaisin kytkimelle. /1/

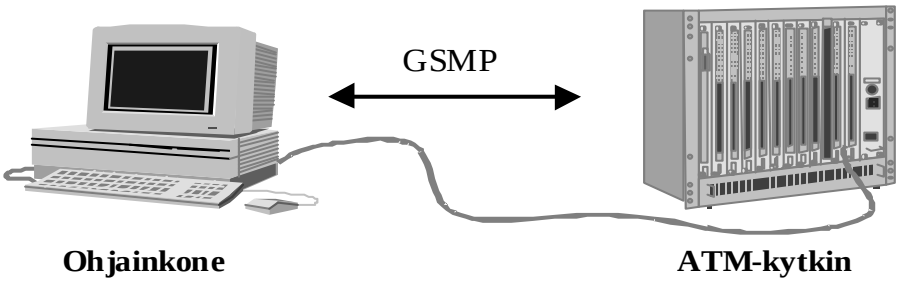
5 KYTKENTÄKENTÄN OHJAUS

Kytkentäkentän ohjauksella tarkoitetaan kytkimen (laitteiston) ohjausta. Ohjaukseen kuuluu fyysisten kytkentöjen muodostaminen ja purkaminen sekä laitteiston tilan seuranta. Useimmiten kytkinlaitteistossa on kytkentäkentän lisäksi sisäänrakennettu ohjain ohjausohjelmistoineen. Kytkentäkentän ohjausta varten eri laitteistovalmistajat ovatkin määritelleet omat rajapintansa. Näin käyttäjä (operaattori) näkee useimmiten vain käyttöliittymän kytkimen hallintaohjelmistoon.

Tässä kappaleessa esitellään ATM-kytkimen ohjaukseen määritelty yleinen protokolla sekä lyhyesti VTT:n (Valtion teknillinen tutkimuskeskus) FSR-kytkimen (Frame Synchronized Ring) ohjausrajapinta.

5.1 GSMP

GSMP (General Switch Management Protocol) on Ipsilonin (nykyisin Nokia Silicon Valley) määrittelemä yleinen ATM-kytkimen ohjaus-protokolla. Siitä on julkaistu kaksi IETF:n (Internet Engineering Task Force) RFC:tä (Request For Comments): versio 1.1 (RFC 1987) sekä tämän laajennus versio 2.0 (RFC 2297 /10/). GSMP:n avulla voidaan helposti erottaa kytkimen ohjausohjelmisto ja kytkentäkenttä toisistaan.



Kuva 12. Ohjauskoneen erottaminen kytkentäkentästä.

5.1.1 Johdanto

GSMP mahdollistaa kytkimen ohjaimen muodostaa ja purkaa yhteyksiä kytkentäkenttään. Sen avulla voidaan lisätä haaroja ryhmäjakeluyhteyksiin (multicast), hallita kytkimen portteja ja kysyä sekä konfigurointi- että tilastotietoja kytkimeltä. GSMP sallii myös kytkimen informoida ohjainta kytkimen tapahtumista asynkronisilla viesteillä. GSMP on asymmetrinen protokolla ohjaimen toimiessa isäntänä ja kytkimen ollessa orjana. Näiden välille muodostetaan kontrolliyhteys ATM-linkin yli ohjausohjelmiston alustusvaiheessa. Yhdellä ohjaimella voidaan ohjata useita kytkimiä. Kontrolliyhteydelle käytetään oletuksena virtuaalikanavaa VPI=0, VCI=15. Versio 2.0 määrittelee GSMP-operaatiot myös Ethernet-verkon yli. /10/.

5.1.2 Rakenne

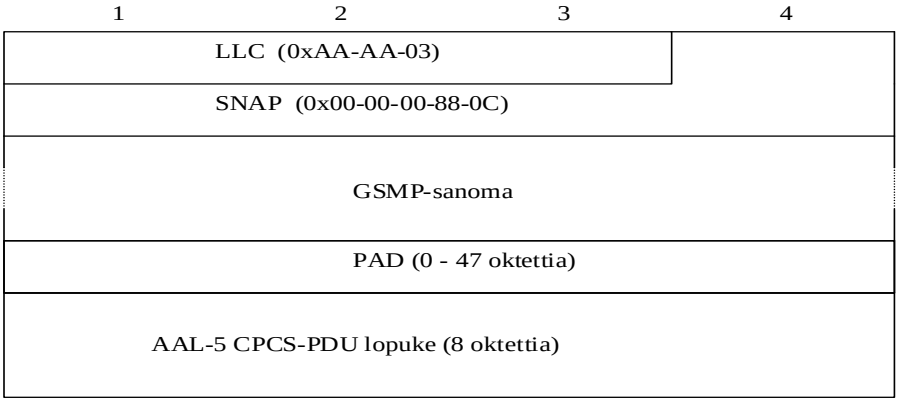
GSMP-protokolla vaatii vakaan kontrolliyhteyden. Se sisältää erillisen synkronointiprotokollan (adjacency protocol), jonka tehtävänä on synkronoida kontrolliyhteys ohjaimen ja kytkimen välillä. Tämän protokollan avulla selvitetään myös käytettävä GSMP-protokollaversio, todetaan linkin toisen pään identiteetti ja havaitaan sen muuttuminen. Yhtään varsinaista GSMP-sanomaa ei lähetetä ennen kuin kontrolliyhteys on synkronoitu. Synkronointiprotokolla tarkistaa linkin synkronoinnin säännöllisin väliajoin /10/.

5.1.3 Pakettien kapselointi

GSMP-paketit ovat vaihtelevan mittaisia ja ATM:n datalinkkikerrosta varten ne kapseloidaan suoraan AAL-5 CPCS-PDU:hun (Common Part Convergence Sublayer) käyttäen LLC/SNAP (Logical Link Control/ SubNetwork Attachment Point) otsaketta.

SNAP-otsikon kolme ensimmäistä oktettia muodostavat OUI-kentän (Organizationally Unique Identifier) ja sen arvo 0x000000 ilmoittaa

protokollatunnisteen (PID, Protocol Identifier) olevan Ethernet-typpiä. Protokollatunnisteen arvo 0x880C on varattu GSMP:lle.



Kuva 13. GSMP-paketin kapselointi AAL-5 CPCS-PDU:hun.

5.1.4 Sanomat

GSMP määrittelee sanomille viisi luokkaa, joilla ohjain tekee pyyntöjä kytkimelle ja saa siltä vastauksen. Nämä ovat yhteydenhallinta-, portinhallinta-, tila- ja tilasto-, konfiguraatio- sekä QoS-sanomat. Näiden lisäksi kytkin lähettää tapahtumasanomia, joilla informoidaan ohjainta asynkronisista tapahtumista.

Yhteydenhallintasanomat

Ohjain käyttää yhteydenhallintasanomia muodostamaan, poistamaan, muuttamaan ja tarkistamaan virtuaalikanava- ja virtuaalipolkuyhteyksiä kytkentäkentän läpi. GSMP 2.0 tukee pisteestä pisteeseen (p-p, point-to-point) ja pisteestä moneen pisteeseen (p-to-mp, point-to-multipoint) yhteyksiä. Monesta pisteestä pisteeseen (mp-to-p, multipoint-to-point) yhteys määritellään luomalla useita p-to-p yhteyksiä, joilla on sama ulostulohaara. Monesta pisteestä moneen pisteeseen (mp-to-mp, multipoint-to-multipoint) yhteys määritellään vastaavasti luomalla useita p-to-mp yhteyksiä, joilla on samat ulostulohaarat.

Taulukko 1. Yhteydenhallintasanomat.

Sanoma	Käyttö
Add branch	Luodaan virtuaalipolku- tai virtuaalikanavayhteyksiä tai lisätään uusi haara olemassa olevaan yhteyteen. Voidaan käyttää myös tarkistamaan yhteyden tila kytkimessä.
Delete tree	Poistetaan koko virtuaalikanava- tai virtuaalipolkuyhteys eli kaikki yhteyden jäljellä olevat haarat poistetaan.
Verify tree	Tarkistetaan virtuaalikanavayhteyden haarojen määrä. Tämä sanoma on poistettu versio 2.0:sta.
Delete all	Poistetaan kaikki yhteydet kytkimen sisääntuloportista.
Delete branch	Poistetaan yksi tai useampi haara virtuaalikanava- tai virtuaalipolkuyhteydestä. Jos poistettava haara on yhteyden viimeinen, koko poistetaan yhteys.
Move branch	Siirretään haara olemassa olevan yhteyden tämän hetkisestä portista uuteen porttiin.

Portinhallintasanomat

Portinhallintasanomalla voidaan portti ottaa käyttöön tai pois käytöstä, tehdä silmukkakytkentä, palauttaa portti alkutilaan tai muuttaa solusiirtonopeutta. Kaikki nämä toiminnot tehdään samalla portinhallintasanomalla asettamalla sanoman toimintakenttään sopiva arvo.

Taulukko 2. Portinhallintasanomat.

Sanoma	Käyttö
Bring Up	Ota portti käyttöön.
Take Down	Ota portti pois käytöstä.
Internal Loopback (sisäinen takaisinkytkentä)	Kytkentäkentältä ulostuloporttiin saapuvat solut ohjataan sisäänmenoportin kautta takaisin kytkentäkenttään.
External Loopback (ulkoinen takaisinkytkentä)	Ulkoisesta linkistä sisääntuloporttiin saapuvat solut ohjataan välittömästi takaisin linkille fyysisellä tasolla.
Bothway Loopback	Sekä sisäinen että ulkoinen takaisinkytkentä suoritetaan.
Reset Input Port	Kaikki tiettyyn sisääntuloporttiin saapuvat yhteydet poistetaan ja sekä sisääntulo- että

	ulostuloportti alustetaan uudelleen.
Reset Event Flag	Nollataan portin halutut tapahtumaliput
Set Transmit Cell Rate	Asetetaan ulosmenoportin solusiirtonopeus mahdollisimman lähelle haluttua arvoa.

Lisäksi on oma sanomansa (label range message) porttien VPI/VCI oletusarvoalueiden muuttamiseen. Kytkimen alustusvaiheessa porteille on asetettu minimi- ja maksimioletusarvot jollakin GSMP:stä riippumattomalla tavalla.

Tila- ja tilastosanomat

Tila- ja tilastosanomat sallivat ohjaimen pyytää erilaisten sisääntulo- ja ulostuloportteihin, virtuaalikanava- ja virtuaalipolkuyhteyksiin sekä palvelunlaatuun sidottujen laitteistolaskureiden arvoja. Niitä käytetään myös kytkimen sisääntuloporttien tilatietojen kysymiseen.

Taulukko 3. Tila- ja tilastosanomat.

Sanoma	Käyttö
Connection Activity	Selvitetään onko yhdellä tai useammalla virtuaalikanava- tai virtuaalipolkuyhteydellä tällä hetkellä liikennettä.
Port Statistics	Pyydetään portin tilastotiedot.
Connection Statistics	Pyydetään virtuaalikanava- tai virtuaalipolkuyhteyden tilastotiedot.
QoS Class Statistics	Pyydetään tietyn portin QoS-luokan tilastotiedot.
Report Connection State	Pyydetään sisääntuloporttia raportoimaan yksittäisen virtuaalikanava- tai virtuaalipolkuyhteyden, tai portin kaikkien yhteyksien tilat.

Konfiguraatiosanomat

Konfiguraatiosanomilla ohjain voi selvittää kytkimen kokoonpanon. Mahdollisia tietoja ovat mm. kytkimen nimi-, tyyppi- ja versiotiedot sekä porttien kapasiteetit.

Taulukko 4. Konfiguraatiosanomat.

Sanoma	Käyttö
Switch Configuration	Pyydetään kytkimen tiedot (ei porttikohtainen).
Port Configuration	Pyydetään yksittäisen portin konfiguraatiotiedot.
All Ports Configuration	Pyydetään kytkimen kaikkien porttien konfiguraatiotiedot.

Tapahtumasanomat

Tapahtumasanomat sallivat kytkimen informoida ohjainta tietyistä asynkronisista tapahtumista. Ohjain ei lähetä kuittausta tapahtumasanomiin.

Taulukko 5. Tapahtumasanomat.

Sanoma	Käyttö
Port Up	Tietyn portin tila on vaihtunut käytössä-tilaan.
Port Down	Tietyn portin tila on vaihtunut pois käytöstä –tilaan.
Invalid VPI/VCI	Yksi tai useampia soluja on saapunut sisääntuloporttiin sisältäen varaamattoman VPI/VCI-arvoparin.
New Port	Uusi portti lisätty kytkimeen.
Dead Port	Portti poistettu kytkimestä.

QoS-sanomat

GSMP:n QoS-sanomat sallivat ohjaimen ryhmitellä virtuaalipolku- ja virtuaalikanavayhteyksiä QoS-luokkiin ja varata QoS-resursseja sekä QoS-luokille että yksittäisille yhteyksille.

Taulukko 6. QoS-sanomat.

Sanoma	Käyttö
Qos Configuration	Ohjain selvittää kytkimen QoS-ominaisuudet.
Scheduler Establishment	Konfiguroidaan tietyn ulostuloportin solujakaja, joka jakaa ulostulolinkin

	kaistan QoS-luokkien ja yksittäisten yhteyksien kesken.
Qos Class Establishment	Konfiguroidaan QoS-luokka tietyssä portissa tai muutetaan jo muodostetun QoS-luokan parametreja.
Qos Release	Poistetaan solujakajan tunniste tai QoS-luokan tunniste ja vapautetaan kaikki siihen liittyvät resurssit.
QoS Connection Management	Luodaan tai muutetaan virtuaalipolku- tai virtuaalikanavayhteyksiä, jotka vaativat määriteltyjä QoS-parametreja.

QoS Connection Management –sanoman toiminta on vastaava kuin Add Branch –sanomalla (yhteydenhallintasanoma). Yhteydenpurkua varten ei ole määritelty erillistä sanomaa QoS-yhteyksille, vaan käytetään Delete Tree, Delete All ja Delete Branch –sanomia.

5.2 FSR-API

FSR-kytkin perustuu rengasväylään, johon liitetään ATM-rajapinnan toteuttavia portteja. Tällä hetkellä tuetaan kahdentyyppisiä kuituliitännöillä varustettuja rajapintoja: 155 Mbps SDH/STM-1 (Synchronous Digital Hierarchy/Synchronous Transport Module) ja 100 Mbps TAXI (Transparent Asynchronous Tranceiver Interface). Ohjaimena voi toimia esimerkiksi Pentium-pohjainen PC tai SUN-työasema. FSR-API (Application Programming Interface) tukee Solaris- ja Linux-käyttöjärjestelmiä. Ohjainkone kytketään FSR:ään ATM-linkillä (SDH/STM-1). /11/

Jokainen portti sisältää mikro-ohjaimen, joka ylläpitää portin rekistereitä. Ohjainkone lähettää alhaisen tason FSR-komentoja portin rajapintaan aktivoidakseen merkinantopyyntöjä kytkimessä. Portti voidaan ohjelmoida myös lähettämään tilasto- ja hälytystietoja ohjaimelle. /11/

FSR-API tarjoaa käyttäjälle selkeän ja yksinkertaisen C-kielisen ohjelmointirajapinnan. Rajapinta on laitteistoriippumaton, joten ohjelmointi ei

vaadi FSR-kytkimen syvällisempää tuntemusta. FSR-API tarjoaa funktiot sekä FSR-kytkimen tilan kontrollointiin että pisteestä-pisteeseen ja pisteestä-moneen pisteeseen yhteyksien luomiseen ja purkamiseen. FSR-API:n tarkoituksena on tukea kaikkia kytkinlaitteiston tarjoamia operaatioita. /11/

6 OLIOMENETELMÄT

Oliotekniikat alkoivat konkretisoitua 80-luvulla. Tutkimuslaboratoriot tuottivat oliopohjaiset ohjelmointikielet: Smalltalk:n ja C++:n. Suunnittelumenetelmät yleistyivät teollisuudessa 70- ja 80-luvuilla. Tekniikoita, jotka mahdollistivat hyvän analyysin ja suunnittelun, pidettiin yhtä tärkeinä kuin ohjelmointikielten oliopohjaista kehitystä.

Tässä kappaleessa esitellään mallinnuksen, suunnittelun ja protokollatoteutuksen olioperustaisia apuvälineitä: mallinnuskieli, suunnittelumallit, oliohajautus sekä protokollaohjelmointiin kehitetty sovelluskehys.

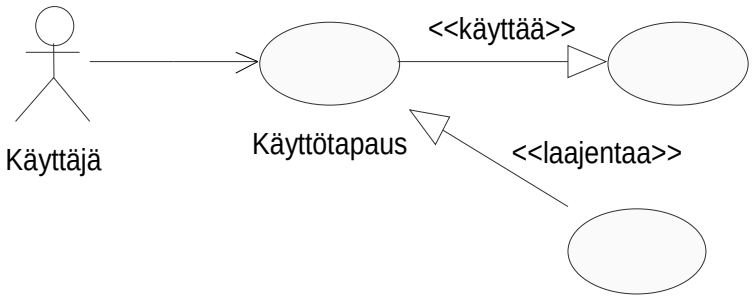
6.1 UML

UML (Unified Modeling Language) on 80- ja 90-luvun vaihteessa ilmestyneiden oliosuuntautuneen analyysin ja suunnittelumenetelmien seuraaja. Erilaisten oliomallien ja -notaatioiden kirjo oli runsas. Jokaisella menetelmällä oli omat käsitteensä ja kuvaustekniikkansa. UML syntyi, kun kolme olioperustaisen mallintamisen kehittäjää yhdistivät menetelmänsä: Jim Rumbaughin OMT (Object Modeling Technique), Ivar Jakobssonin OOSE (Object-Oriented Software Engineering) ja Grady Boochin BOOCH. Tavoitteena heillä oli luoda yleinen standardoitu mallinnuskieli, joka rohkaisisi ja helpottaisi suunnittelijoita mallintamaan ohjelmistonsa kunnolla ennen toteutusta. OMG (Object Management Group) on standardoinut UML-kielen.

Mallintamisella pyritään todellisuuden yksinkertaistamiseen. Mallintamisen avulla pystytään helpommin ymmärtämään monimutkaisia järjestelmiä. UML on nimenomaan mallinnuskieli. Se määrittelee mallinnuksen (pääasiassa) graafiset esitysmuodot. UML:ää käytetään ohjelmistojen ja

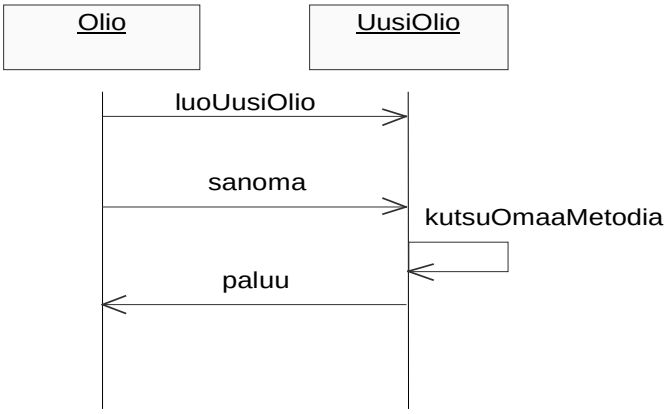
liiketoimintaprosessien visualisointiin, määrittelyyn, toteutukseen ja dokumentointiin. /12/

UML tarjoaa tuen kaikkiin kehitysvaiheisiin, joita kuvataan erilaisilla kaavioilla. Käyttötapauskaavio (use case diagram) esittää käyttäjän ja käyttötapauksen väliset suhteet kuvaten järjestelmän käyttöä. Käyttötapaus (use case) erottaa jonkin toiminnallisen kokonaisuuden systeemissä. Se on käyttäjän alulle panema ja tuottaa jonkin tuloksen. Käyttäjä (actor) on esimerkiksi henkilö tai laite, joka käyttää järjestelmää. Käyttötapausten selvittäminen on ohjelmistomäärittelyn ensimmäinen vaihe. /12//13/



Kuva 14. Käyttötapauskaavio.

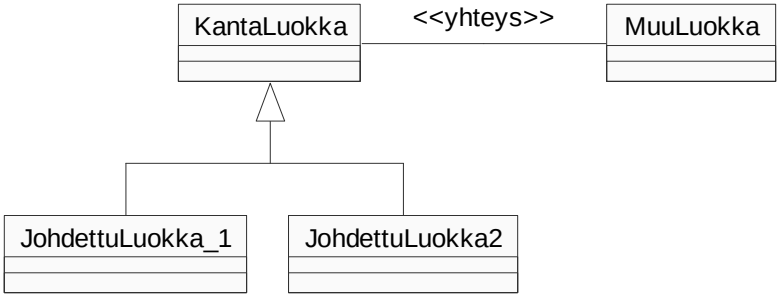
Olioiden yhteistoimintaa kuvataan yhteistoimintakaavioilla. Ne kertovat kuinka olioiden palveluita käytetään. UML määrittelee yhteistoiminnalle kaksi kuvaustapaa: sekvenssikaavion (sequence diagram) ja olioyhteistyö-kaavion (collaboration diagram). Sekvenssikaavio kuvaa olioiden väliset operaatiot ja vastaukset aikajärjestyksessä.



Kuva 15. Sekvenssikaavio.

Olioyhteistyökaavio ei keskity aikajärjestykseen, vaan se kuvaa olioiden välisten kytkentöjen hyödyntämistä olioiden yhteistoiminnassa.

Luokkakaavio (class diagram) on olioperustaisen suunnittelun keskeisin tekniikka. Sillä kuvataan systeemin staattinen rakenne. Luokkakaaviolla esitetään systeemin olioiden tyypit ja niiden väliset staattiset suhteet. Kaaviosta selviää luokkien perintähierarkia sekä luokkien väliset kytkentätyypit. Luokkakaaviossa voidaan esittää myös luokan attribuutit ja operaatiot.



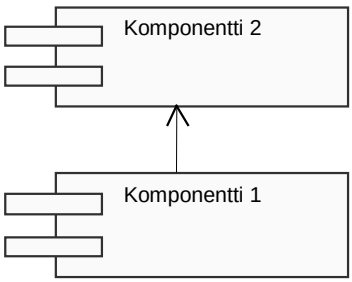
Kuva 16. Luokkakaavio.

Tilakaavio (statechart diagram) on tunnettu menetelmä kuvaamaan systeemin käyttäytymistä. Sillä kuvataan elementin (useimmiten luokka) tilat ja tilavaihdokset tiettyjen tapahtumien seurauksena. Tilakaaviolla

esitetään useimmiten yksittäisen olion käyttäytymistä useissa käyttötapauksissa.

Aktiviteettikaavio (activity diagram) on muunnelma tilakaaviosta. Se soveltuu hyvin työnkulun mallintamiseen ja rinnakkaisprosessoinnin kuvaamiseen. Aktiviteettikaaviota voidaan käyttää mm. monisäieohjelmointiin.

Toteutuskaavioilla esitetään ohjelmiston toteutukseen liittyviä näkökohtia. Komponenttikaavio (component diagram) kuvaa itse koodin rakennetta ja käyttökaavio (deployment diagram) ilmaisee systeemin ajoaikaisen rakenteen sisältäen komponentteja, prosesseja ja niiden aikaisia olioita. Ohjelmistomoduulit kuvataan komponenttityyppeinä.



Kuva 17. Komponenttikaavio.

Pakettikaavio (package diagram) on tärkeä menetelmä, jolla laaja ohjelmistosysteemi jaetaan pienempiin, helpommin ymmärrettäviin osiin. Kaikki UML:n mallinnuselementit voidaan organisoida paketeiksi.

6.2 Suunnittelumallit

”The Gang of Four” (GoF – Gamma, Helm, Johnson ja Vlissides) julkaisi vuonna 1995 kirjan nimeltä Design Patterns /14/, jossa luetteloidaan ja dokumentoidaan kokemuksia olioperustaisesta ohjelmistosuunnittelusta. Kirjassa listataan yleisimmät suunnittelumallit (design patterns).

Suunnittelumallit ovat helposti sovellettavia, hyvin dokumentoituja ja yksinkertaisia ratkaisuja spesifisiin ongelmiin olioperustaisessa ohjelmisto-suunnittelussa.

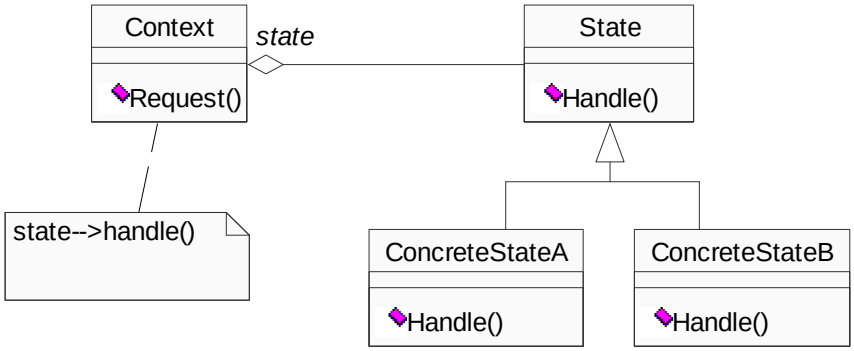
Suunnittelumallilla on neljä rakenneosaa: nimi, ongelma, ratkaisu ja tulos. Suunnittelumallin nimi kuvaa ongelman, ratkaisun ja tuloksen sanalla tai kahdella. Se helpottaa dokumentointia ja ryhmätyöskentelyä kertomalla mistä suunnittelumallista on kyse. Ongelma kertoo, milloin ko. suunnittelumallia voidaan käyttää. Ratkaisu kuvaa mallin elementit (luokat tai oliot), niiden väliset suhteet ja yhteistoiminnan. Tulokset kertovat suunnittelumallin käytön seuraamukset, kuten vaikutukset systeemin joustavuuteen, laajennettavuuteen ja siirrettävyyteen. /14/

Suunnittelumallit jaetaan kolmeen ryhmään. Olioiden luomiseen liittyvät mallit (creational patterns) auttavat tekemään systeemistä olioiden luonti-tavasta riippumattoman. Rakenteelliset mallit (structural patterns) keskit-tyvät siihen, kuinka luokat ja oliot muodostavat suurempia rakenteita. Olioiden käyttäytymiseen liittyvät mallit (behavioral patterns) keskittyvät algoritmeihin ja olioiden väliseen kommunikointiin.

Taulukko 7. Suunnittelumallit ryhmittäin.

Creational Patterns	Structural Patterns	Behavioral Patterns
Abstract Factory	Adapter	Chain of Responsibility
		Command
Builder	Bridge	Interpreter
		Iterator
Factory Method	Composite	Memento
		Observer
Prototype	Decorator	State
	Facade	Strategy
Singleton	Flyweight	Template Method
	Proxy	Visitor

Oheinen esimerkkikuva esittää State-suunnittelumallin rakennetta. Tätä suunnittelumallia voidaan käyttää mm. protokollaohjelmoinnissa protokollan tilakoneen toteuttamiseen. Tällöin jokaista tilaa varten luodaan oma luokkansa (concrete state), ja protokollan (context) tilamuuttuja osoittaa aina oikeaan tilaolioon. Suunnittelumalleja käytetään usein rinnakkain, esimerkiksi state-suunnittelumallin tilat voidaan toteuttaa singleton-suunnittelumallin mukaisesti.



Kuva 18. State-suunnittelumalli.

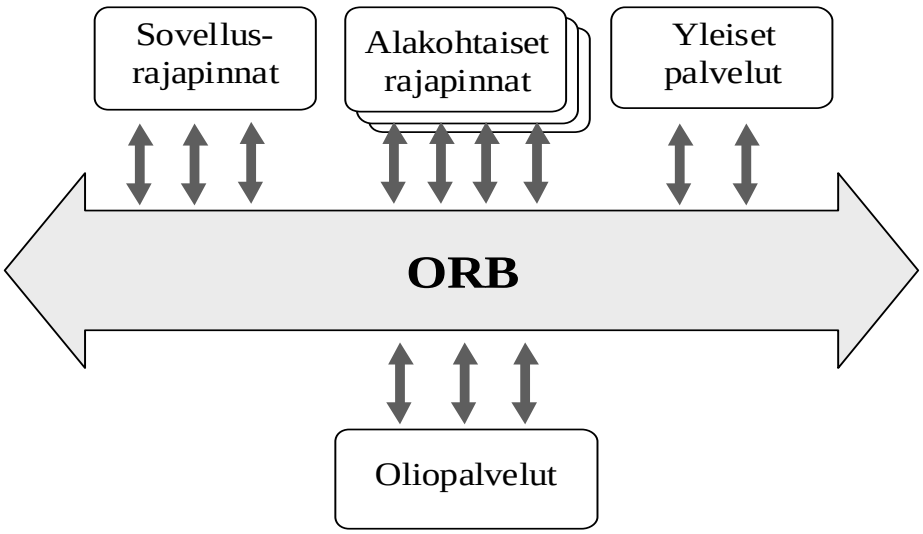
6.3 CORBA

CORBA (Common Object Request Broker Architecture) on OMG:n ratkaisu tukea ohjelmistojen yhteistoimintaa ja siirrettävyyttä heterogeenisessa ohjelmisto- ja laiteympäristöissä. OMG perustettiin vuonna 1989 kehittämään ja edistämään standardointia hajautetun heterogeenisen ympäristön ohjelmistokehitystä varten. Nyt OMG on kasvanut jo yli 800:lla jäsenyrityksellään maailman suurimmaksi ohjelmistojärjestöksi. /15/

6.3.1 Arkkitehtuuri

OMA (Object Management Architecture) on OMG:n määrittelemä oliomallin ja viitemallin yhdistelmä. Oliomalli määrittelee kuinka

heterogeeniseen ympäristöön hajautetut oliot voidaan kuvata. Viitemalli puolestaan kuvaa kuinka nämä oliot kommunikoivat. /15/



Kuva 19. OMA-viitemallin komponentit.

OMA-viitemallissa on määritetty neljä ryhmää oliorajapinnoille sekä ORB (Object Request Broker) eli oliokutsujen välittäjä. Oliopalvelut (object services) ja yleiset palvelut (common facilities) ovat alasta riippumattomia palveluita. Oliopalvelut, kuten nimipalvelu (naming service) ja tapahtuma-palvelu (event service), ovat usein käytettyjä CORBA:n peruspalveluita. Yleiset palvelut (common facilities) ovat käyttäjälle näkyviä korkean tason sovelluspalasia. Alakohtaisia rajapintoja (domain interfaces) on määritelty tietoliikenteen lisäksi mm. lääketieteen ja kaupan aloille. Sovellusraja-pinnat (application interfaces) ovat käyttäjän määrittelemiä sovellus-kohtaisia rajapintoja, jotka eivät kuulu OMG:n standardoinnin piiriin. /15/

6.3.2 ORB

ORB on välisovellus (middleware), joka tarjoaa mekanismin läpinäkyvään kommunikointiin asiakasolion ja kohdeolion välillä. ORB välittää asiakkaan (client) kutsut ja parametrit sekä vastaukset asiakasoliolle. Asiakkaan ei

tarvitse tietää kohdeolion sijaintia, toteutuskieltä, käyttöjärjestelmää tai muutakaan systeemiin liittyvää näkökohtaa. Kohdeolio voi olla paikallinen tai sijaita eri koneessa. CORBA-olioon viitataan olioviitteen avulla, joka luodaan CORBA-olion luonnin yhteydessä. Olioviite saadaan luomalla uusia olio tai hakemalla se nimipalvelusta (naming service) nimikontekstin avulla tai ’kauppapalvelusta’ (trading service) olion ominaisuuden perusteella. ORB sisältää myös metodit, joilla olioviite voidaan muuntaa merkkijonoksi ja päinvastoin. Tämän ominaisuuden avulla olioviite voidaan tallettaa helposti tiedostoon tai tietokantaan. /15/

Aluksi eri ORB-valmistajien toteutukset eivät olleet yhteensopivia keskenään. Vasta CORBA 2.0 spesifikaatiossa esitellään arkkitehtuuri yhteentoimivuutta (interoperability) varten. CORBA 2.0 määrittelee GIOP-protokollan (General Inter-ORB Protocol) ja sen sovituksen TCP/IP-protokollapinoon. TCP/IP-sovitusta kutsutaan IIOP-protokollaksi (Internet Inter-ORB Protocol). Nämä kaksi protokollaa ovat pakollisia CORBA-yhteensopiville toteutuksille ja ne määrittelevät standardin formaatin olioiden väliseen kommunikointiin.

6.3.3 Rajapinnan kuvauskieli

CORBA-olioiden rajapintojen kuvaamiseen käytetään IDL-kieltä (Interface Definition Language). Se on OMG:n määrittelemä rajapinnankuvauskieli, joka on syntaksiltaan hyvin samanlainen kuin C++ tai Java. IDL-kieli on määrittelykieli, jonka avulla kuvataan vain olioiden rajapintoja, ei toteutusta. Se on ohjelmointikielestä riippumaton. Kieliriippumattomuus mahdollistaa olioiden toteuttamisen eri ohjelmointikielillä sekä olioiden välisen kommunikoinnin heterogeenisessa ympäristössä. Rajapinnoissa kuvataan attribuutit, tietotyypit, poikkeustilanteet (exception) sekä operaatiot parametreineen ja paluuarvoineen. Rajapinnat voidaan ryhmitellä moduuleiksi. IDL-kieli määrittelee perustietotyypit (esim. long, double,

boolean), rakenteiset tyypit (struct ja union) sekä templaattityypit (esim. sequence ja string). /15/

OMG on määritellyt IDL:lle kielisidonnan useille eri ohjelmointikielille (esim. C, C++, Java, Smalltalk ja Ada). Kielisidonta tarkoittaa rajapinnan kuvautumista ko. ohjelmointikielelle. Esimerkiksi C++-kielessä IDL-rajapinta kuvautuu luokaksi ja rajapinnan operaatiot luokan metodeiksi. IDL-rajapinta muunnetaan toteutuskielelle erillisellä IDL-kääntäjällä ennen CORBA-olion toteutusta.

6.4 OVOPS++

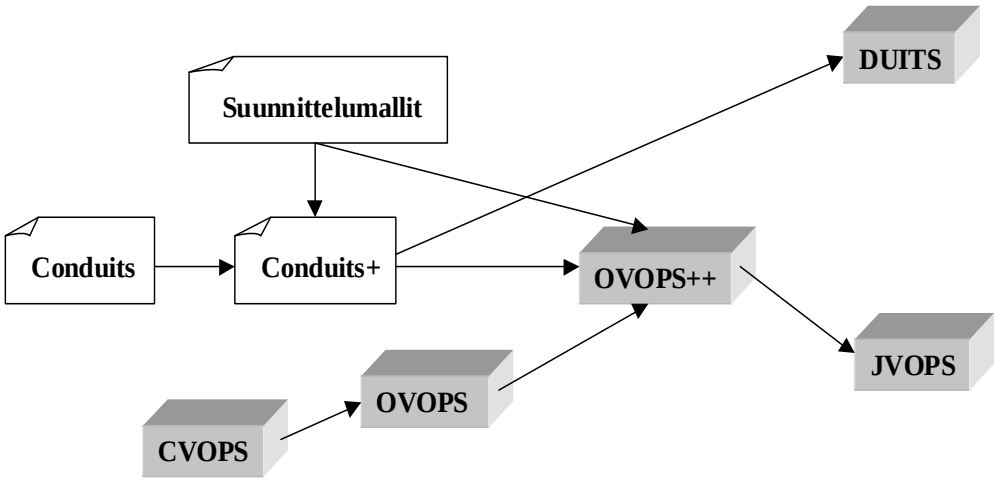
OVOPS++ (Object Virtual Operating System ++) on sovelluskehys (framework) kerrosmalliin perustuvien tietoliikenneprotokollien toteutukseen. OVOPS++ kehitettiin Teknillisen korkeakoulun (TKK) TOVE-projektissa (Toimialaverkot).

6.4.1 Johdanto

OVOPS++ on toteutettu C++-kielellä ja se perustuu Conduits+-sovelluskehyksestä julkaistuun dokumenttiin /16/ sekä joihinkin OVOPS-ympäristöstä lainattuihin ideoihin. Conduits+ on oliopohjainen, vahvasti suunnittelumalleihin perustuva sovelluskehys tietoliikenneprotokollien toteuttamiseen. Sen ovat kehittäneet yhdessä Ascom Tech AG, Illinoisin yliopisto ja GLUE Software Engineering /17/.

OVOPS on Telecom Finland:n (nykyisin Sonera) ja Lappeenrannan teknillisen korkeakoulun (LTKK) kehittämä hajautettujen ohjelmistojen kehittämisympäristö. OVOPS suunniteltiin tukemaan laajojen oliopohjaisten sovelluksien toteutusta. Se on saanut runsaasti vaikutteita VTT:n kehittämästä CVOPS-protokollatyökalusta (C-based Virtual Operating System). /18/

Nykyisin on toteutettu myös Java-perustaisia Conduit+-idean mukaisia sovelluskehyksiä.



Kuva 20. OVOPS++-sovelluskehyksen historia.

6.4.2 Rakennuselementit

OVOPS++:n rakennuselementtejä (conduit) on neljänlaisia: protokolla (protocol), multiplekseri (mux), tehdas (factory) ja sovitin (adapter). Rakennuselementti on ohjelmistokomponentti, jolla on kaksi sivua: A- ja B-puoli. Rakennuselementin sivuihin voidaan kytkeä mikä tahansa toinen rakennuselementti. Rakennuselementit ovat kaksisuuntaisia ja ne kommunikoivat toistensa kanssa sanomien avulla.

Protokollaelementtiä käytetään toteuttamaan yksi protokollapinon kerros. Se sisältää tilakonemekanismin ja käsittelee sanomia oletuksena asynkronisesti. Protokollaelementtiin on kytkettynä yksi rakennuselementti sen kummallekin puolelle. Multiplekseri on rakennuselementti, jolla on yksi A-puoli ja useita B-puolia. Se kokoaa B-puolilta tulevat sanomat A-puolelle, ja jakaa A-puolelle tulevat sanomat oikealle B-puolelle sanomassa olevan avaimen avulla. Tehdaselementtiä käytetään luomaan dynaamisesti uusia rakennus-

elementtejä multiplekserin B-puolelle. Tehdaselementtiin alustetaan prototyyppi luotavista rakennuselementeistä. Tehdas kloonaa prototyyppiä luodessaan uusia rakennuselementtejä. Sovitinelementtiin ei ole B-puolelle kytkettynä toisia rakennuselementtejä, vaan se toimii sovelluskehyksen ja jonkin toisen ohjelmiston tai laitteen välisenä rajapintana. Sovitin voi olla esimerkiksi laitteistorajapinta tai CORBA-sovitin.

6.4.3 Sanomat

Protokollakerrosten väliset primitiivit ja vastinolioiden väliset PDU:t peritään sanomaoliosta. OVOPS++ tarjoaa primitiiveissä kuljetettavalle protokollainformaatiolle kehysrakenteen (frame), johon PDU koodataan. OVOPS++ sisältää myös rakenteen PDU:lle ja informaatioelementille koodereineen. Sanoman ominaisuuksiin kuuluu lisäksi tieto siitä, mitä tilakoneen toimintametodia se kutsuu saapuessaan protokollaan.

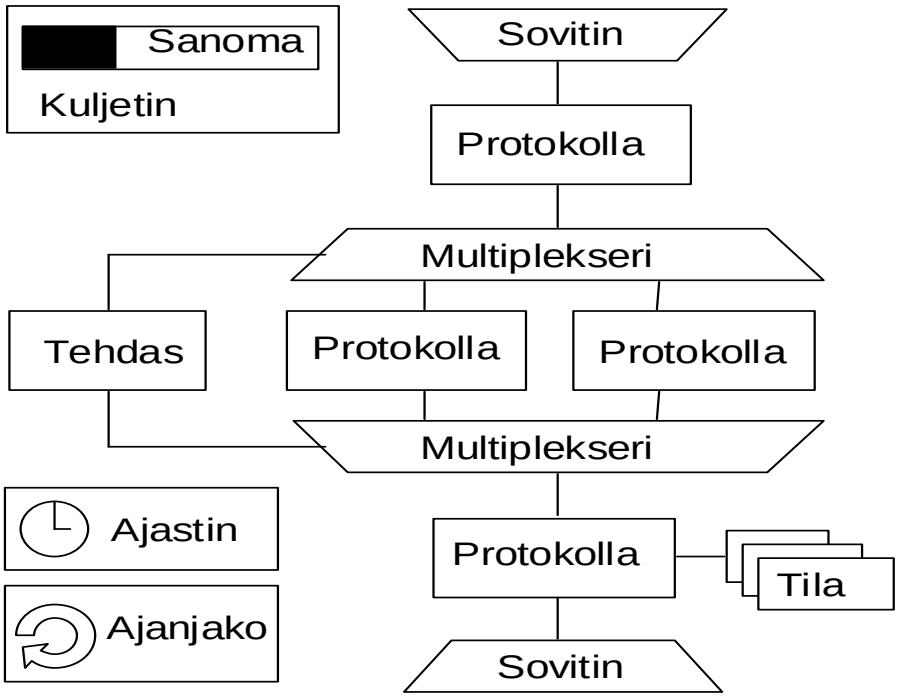
Tilakone ja sanomat ovat protokollaspesifisiä ominaisuuksia. Jotta informaatiota voidaan kuljettaa kaikkien rakennuselementtien läpi, tarvitaan kuljetin (transporter). Kuljetin on visitor-suunnittelumallin mukainen, ja se tietää kuinka toimia kunkin tyyppisessä rakennuselementissä. Sovelluskehyksessä on valmiina sanomakuljettimen (message transporter) lisäksi useita erikoistehtäviin, kuten protokollien asentamiseen ja purkamiseen, suunniteltuja kuljettimia.

6.4.4 Ajanjako

Ajanjakosysteemi (scheduler) perustuu OVOPS-ympäristön ajanjakomekanismiin. Asynkronisesti toimivat rakennuselementit (protokollat) rekisteröityvät ajanjakosysteemiin ja pyytävät sanoman saapuessa suoritusaikaa. Ajanjakosysteemi mahdollistaa erilaisten tapahtumien (kuten CORBA-kutsujen), asynkronisten rakennuselementtien, ajastimien (timer) ja

laitteistojen (esimerkiksi soketit) yhteistoiminnan kontrolloimalla näiden suoritusvuoroja.

Ajastimet voidaan toteuttaa joko erillisinä olioina tai protokollan ominaisuutena. Ajastimelle asetetaan aika ja se voidaan käynnistää tai pysäyttää. Ajastimen lauettua (timeout) se lähettää isännälleen (protokollalle) timeout-sanoman, joka käsitellään normaalisti tilakoneessa.



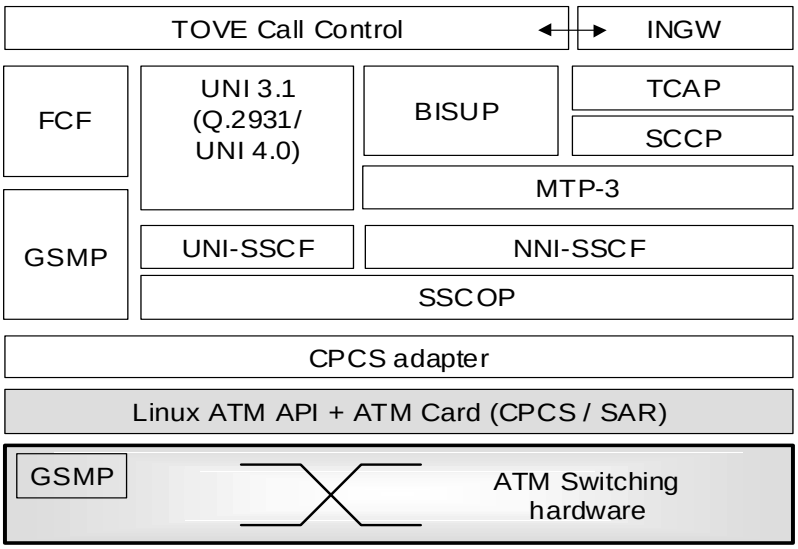
Kuva 21. OVOPS++ -sovelluskehyksen komponenetit.

7 OHJAUSOHJELMISTON TOTEUTUS

Tässä osassa työtä kuvataan ATM-kytkimen ohjausohjelmiston toteutus Teknillisen korkeakoulun TOVE- (Toimialaverkot) ja SCOMS-projekteissa (Software Configurable Multidiscipline Switch). Kuvauksessa keskitytään pääasiassa ohjausohjelmiston rakenteen sekä kytkentään liittyvien olioiden toteutuksen esittämiseen ja vielä toteuttamattomien ideoiden esittelyyn.

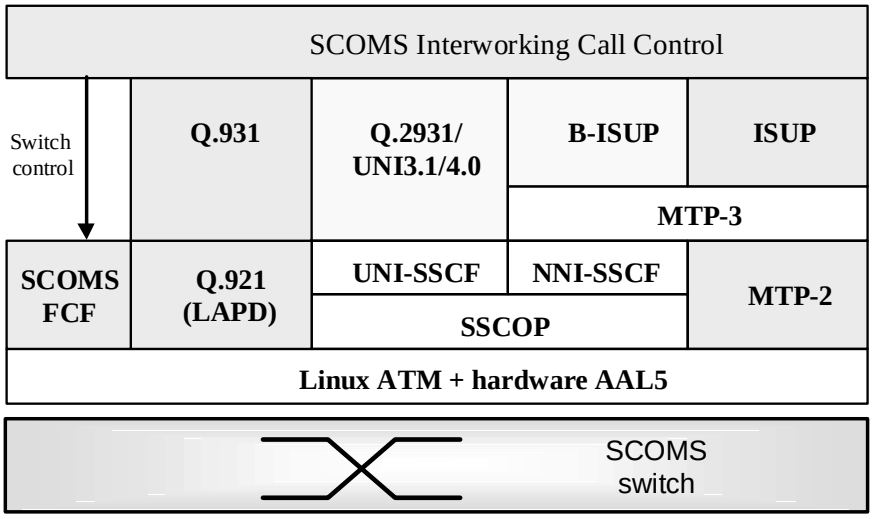
7.1 TOVE- ja SCOMS-projektit

TOVE-projekti on vuosina 1996-1998 toteutettu TEKES:n (Teknologian kehittämiskeskus) ja yrityspartnereiden rahoittama tutkimushanke, jossa kehitettiin tietoliikenneratkaisuja multimediaan pohjautuvaa yritysviestintää varten. Projektissa suunniteltiin ja pilotoitiin ATM-kytkimen ohjausohjelmisto. Ohjelmisto rakentuu OVOPS++-sovelluskehyksen päälle rakennetuista ITU-T:n määrittelemistä laajakaista-ISDN-verkon merkinantoprotokollista, kytkentäkentän ohjauksesta sekä älyverkko-osasta. Kytkentälaitteistona käytettiin VTT:n tarjoamaa FSR-kytkintä.



Kuva 22. TOVE-projektin protokollapinot.

SCOMS-projekti on vuonna 1998 alkanut, vuoden 2000 loppuun kestävä, VTT:n ja TKK:n yhteinen projekti. Siinä kehitetään kytkentään ja reititykseen liittyviä ratkaisuja, jotka tukevat useiden verkkoarkkitehtuurien kytkemistä. Tavoitteena on kehittää verkkosolmu, joka yhdistää piiri-, solu- ja pakettikytkentäiset tekniikat samaan kytkentäkenttään. SCOMS-ohjelmiston runkona käytetään TOVE-projektissa syntynyttä ATM-kytkimen ohjausohjelmistoa. SCOMS-projektissa ohjelmistoa laajennetaan tukemaan 64 kbit/s PDH-kanavien (Plesiochronous Digital Hierarchy) kytkentää, yhteistoimintaa (interworking) ATM- ja PDH-verkkojen välillä sekä kontrolloimaan hajautettua IP-reititystä natiivi-IP-siirtoa tukevilla liityntäkorteilla.

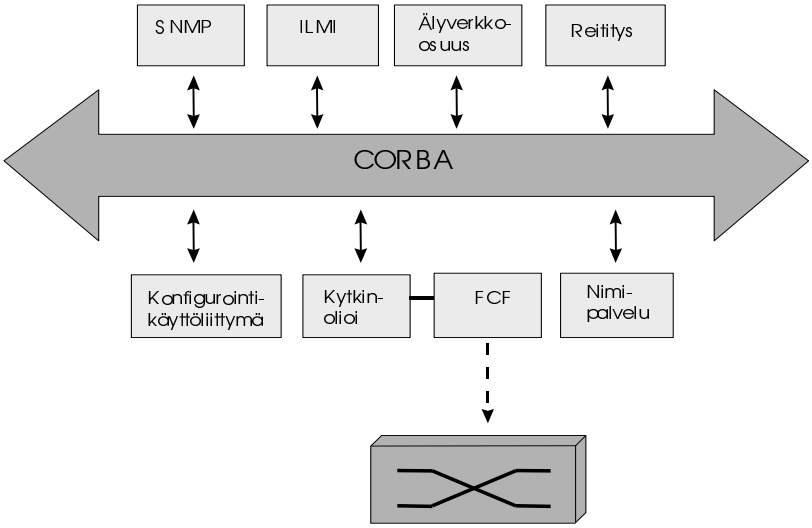


Kuva 23. SCOMS-projektin protokollapinot.

7.2 Hajautettu ohjelmisto

Toteutettu ohjelmistoarkkitehtuuri on hajautettu CORBA:n avulla. Hajautuksella saavutetaan useita etuja. Se tuo ohjelmistoon modulaarisuutta, joka selkeyttää rakennetta ja mahdollistaa verraten helpon laajennettavuuden. Kehitysaikana ohjelmiston kääntäminen ja testaaminen on helpompaa, kun voidaan keskittyä vain haluttuihin ohjelmiston osiin

kerrallaan. Hajautus mahdollistaa tarvittaessa myös prosessoinnin jakamisen eri laitteille, esimerkiksi tietokantoihin liittyvät levyoperaatiot voidaan ohjata erillisen koneen prosessoitavaksi. Lisäksi ORB:ien monipuolinen kielisidonnaisuus mahdollistaa eri ohjelmointikielillä toteutettujen ohjelmiston osien yhteistoiminnan, esimerkiksi merkinantoprotokollat ja kytkinmoduuli on toteutettu C++:lla ja älyverkko-osuus sekä konfigurointikäyttöliittymä Java:lla.



Kuva 24. Hajautettu ohjelmistoarkkitehtuuri.

Kytkimen ohjausohjelmiston ytimenä toimii kytkinolio. Kytkinolio käyttää sopivaa FCF-oliota (Fabric Control Function) kytkentäkentän ohjaukseen. Kytkinolioon on toteutettu useita CORBA-palvelimia (olioita), jotka kommunikoivat hajautettujen moduulien kanssa. Kaikki CORBA-oliot rekisteröidään standardiin nimipalvelimeen, josta palveluja tarvitseva olio voi hakea tarvittavan olioviitteen. Kytkinolion attribuuttien hallintaan määriteltiin yksinkertainen IDL-rajapinta, jota hyödyntävät ILMI, SNMP ja konfigurointikäyttöliittymä. Samaa rajapintaa hyödyntäen voitaisiin toteuttaa mm. erilaisia monitorointi- ja lokikäyttöliittymiä.

Pääosin Javalla toteutetut älyverkko-osuus sekä PNNI:n mukainen reititysmoduuli sisältävät omat IDL-rajapintansa. Puhelunohjaus käyttää suoraan älyverkkototeutusta ja kytkinmoduulin kautta reitityspalvelinta.

7.2.1 ILMI- ja SNMP-agentit

ILMI-sanomien käsittelyä varten toteutettiin kytkinoliosta hajautettu ILMI-agentti. Sen tehtävänä on ASN.1-koodattujen SNMP-sanomien purkaminen ja tarkistus. Lisäksi se toteuttaa tilakoneen ILMI:n tarjoamalle automaattiselle osoitteenrekisteröinnille. Agentti käsittelee kytkinolion attribuutteja (osoitetietoja) CORBA-rajapinnan kautta. Jokaista UNI-linkkiä kohden käynnistetään oma ILMI-agenttinsa.

Myös SNMP-hallintaohjelmistoja varten voidaan toteuttaa erillinen kytkinkohtainen SNMP-agentti, joka käsittelee UDP:n päällä siirrettäviä SNMP-sanomia.

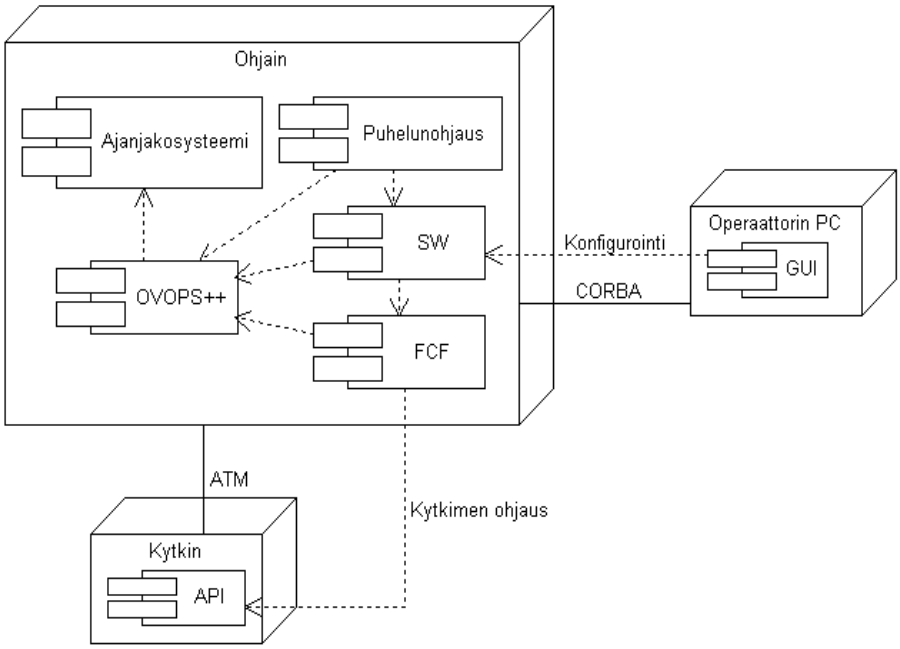
7.3 Kytkinmoduuli

Kytkinmoduuli sisältää koko ohjausohjelmiston ytimenä toimivan kytkinolion, portti- ja linkkioliot, kytkimen ohjausrajapinnan (FCF) sekä edellä mainitut CORBA-palvelimet konfigurointi- ja ILMI/SNMP-pyyntöjen käsittelyyn.

7.3.1 Kytkinolio

Kytkinolio tarjoaa pääsyn systeemin resursseihin. Kytkinolion periaate on facade-suunnittelumallin mukainen, missä kytkinolio tarjoaa yhdistetyn rajapinnan kytkimen resursseihin. Kytkinolio ohjaa systeemin resursseihin liittyvät kutsut asianmukaiselle oliolle (esim. portit, kytkentäkentän ohjaus, ja reititys). Singleton-suunnittelumallin avulla varmistetaan, että kytkimestä on olemassa vain yksi instanssi, ja että sen olioviite saadaan globaalisti. Kytkinoliossa on asetettuna linkkiolioiden ja varsinaisen merkinanto-

protokollan (esim. UNI, BISUP) prototyypit. Linkkejä ja puheluita muodostettaessa prototyyppi kloonataan ja kaikki siihen asetetut parametrit monistuvat luotuihin olioihin. Kytkinolion attribuutteina ovat lisäksi halutun tyyppiset reititys- ja kytkentäkentänohjausoliot, tietorakenne porttien tallentamiseen, SS7-verkon (Signaling System No.7) konfigurointiolio, MTP3-protokolla (Message Transfer Part 3) ja konfigurointirajapinnan toteuttava CORBA-palvelin. Myös MTP3-protokollaan kytketyt käyttäjä-osan ja linkkikerroksen valintaan liittyvät multiplekserit löytyvät kytkin-oliosta.



Kuva 25. UML-käyttökaavio systeemin rakenteesta.

Systeemin laajetessa, ja yhä uusien ominaisuuksien lisääntyessä, ohjausohjelmiston käynnistys monimutkaistuu. Erityisesti liityntä kytkinlaitteeseen sekä CORBA:lla hajautettujen olioiden alustus tuovat synkronointiongelmia. Ratkaisuksi tähän eri moduulien synkronointiin on tilakoneen toteuttaminen kytkinoliolle. Tällöin kytkinoliolle määritellään erikseen alkutila, alustusvaihe ja toimintatila. Myös mahdollisiin

virhetilanteisiin ja kytkimen uudelleenkäynnistykseen voidaan määritellä omat tilansa. /19/

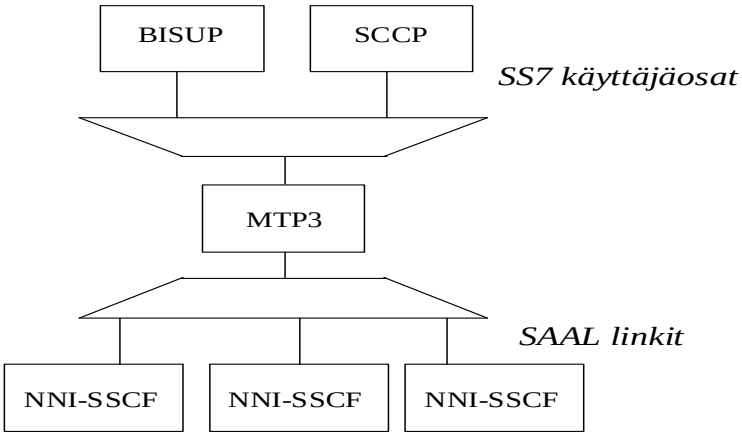
7.3.2 Portit ja linkit

Jokaista kytkinlaitteiston fyysistä porttia vastaa porttiolio. Eri porttityypit rekisteröidään tehtaaseen (factory), jonka avulla porttioliot luodaan ohjelmistoa alustettaessa. Porttiolion tehtävänä on pitää kirjaa kytkimen porttien resursseista, esimerkiksi jokainen ATM-portti ylläpitää tietoa varatuista VPI/VCI-arvoista. CAC:n laajempi toteuttaminen vaatisi ATM-portilta myös kaistan kapasiteetin seurantaa ja erilaisten QoS-ominaisuuksien toteutusta. Ohjausrajapinnasta riippuen osa näistä toiminnoista voisi olla suoraan kytkinlaitteiston vastuulla.

Porttiolioon talletetaan kyseiseen porttiin luodut linkit. Linkki kuvaa loogista merkinantorajapintaa ja se sisältää protokollapinon merkinantoa varten. Linkkiolioiden luomiseen käytetään samaa tehdasta kuin porttiolioidenkin luomiseen. Linkin toteutus on jaettu kahteen luokkaan, jotta eri merkinantopinojen yhtenäiset osat saadaan samaan olio-rakenteeseen. Ylemmät kerrokset sisältävällä linkkioliolla on attribuutteina koordinointiprotokolla, multiplekseri ja tehdas uusien yhteyksien (puheluiden) luomiseen. Linkkioliota luotaessa asetetaan tähän tehtaaseen oikean tyyppisen merkinantoprotokollan (esim. UNI3.1 tai BISUP) prototyyppi. Tehdas luo uusia yhteyksiä (protokollainstansseja) kloonaamalla prototyyppiä ja asettamalla luodun yhteysprotokollan multiplekseriin.

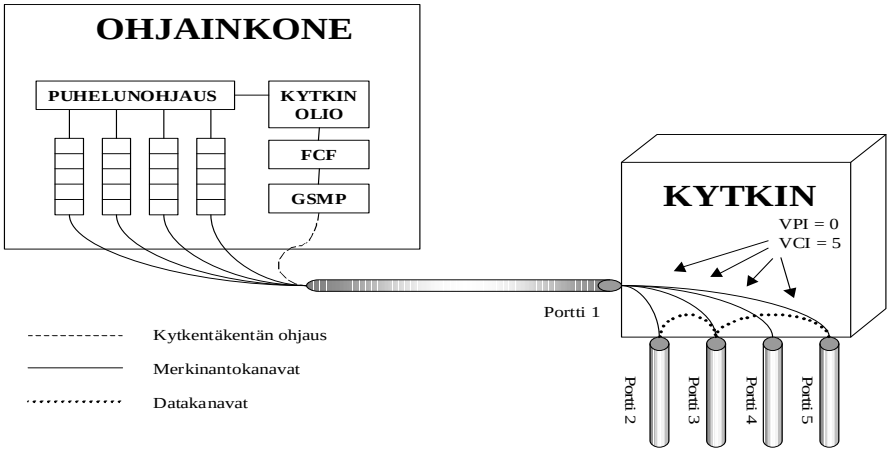
Protokollapinon alemmat kerrokset (ATM:ssä SAAL) talletetaan omaan olioonsa, koska alemman kerroksen linkkejä voi olla eri määrä kuin ylemmän kerroksen olioita. Tämä tulee esille oheisesta kuvasta, jossa MTP3-protokollan kautta eri SS7-verkon käyttäjäosat lähettävät sanomia

samoille linkeille. Liitteessä 1 on luokkakaavio porttien ja linkkien rakenteesta.



Kuva 26. Linkkien jako kahteen osaan SS7-verkossa.

Linkin luominen muodostaa kokonaisuudessaan merkinantopinon laiteraja-pinnan sokettiliitynnästä (ATM:ssä CPCS-sovitin) alkaen multiplekseriin asti eli ennen varsinaista yhteysprotokollaa (esim. UNI tai BISUP). Linkkiä luotaessa FCF:n avulla kytketään kytkinlaitteen tiettyyn liitäntäkorttiin tuleva merkinantokanava ohjainkoneelle, jolloin merkinantoliikenne ohjautuu linkkiolion ylläpitämään protokollapinoon.



Kuva 27. Merkinannon ohjautuminen protokollapinoon.

UNI-linkin tapauksessa porttiolioon luodaan verkonhallintaa varten ILMI-olio, joka käsittelee kyseisen ILMI-yhteyden agentilta tulevat CORBA-kutsut ja välittää ne kytkinoliolle.

7.3.3 Kytkentäkentän ohjaus

Kytkentäkentän ohjaus (FCF) haluttiin toteuttaa omana olionaan erillään puhelunohjauksesta. Sen tehtävänä on peittää laitteistorajapinta ja tarjota yleisempi ohjausrajapinta puhelunohjaukselle. TOVE-projektissa käytettiin kytkimen ohjaukseen GSMP-protokollaa, mutta SCOMS-projektissa on tarvetta käyttää suoraan laitteistotoimittajan ohjelmointirajapintaa (FSR-API). Kytkinolioon voidaan siis helposti alustaa haluttu FCF-olio, joka toteuttaa yleisen ohjausrajapinnan.

FCF:n kautta suoritetaan kytkinlaitteiston merkinantokanavien kytkeminen ohjainkoneelle sekä yhteyksien kytkeminen kytkimen liitäntäkortilta toiselle puhelunmuodostuksessa. Myös edellisiä vastaavat kytkentöjen purku-operaatiot sekä kytkimen parametrien kyselyt suoritetaan FCF:n kautta. Kytkinlaitteiston sekä sen ohjausrajapinnan tukemista ominaisuuksista riippuen, voidaan FCF:ään toteuttaa erilaisia tilatietojen ja liikenteen seurannan funktioita. Kovin yksityiskohtaiset ominaisuudet tosin vaikeuttavat rajapinnan yleisyyden säilyttämistä.

Liitteessä 2 on esimerkki (UML-sekvenssikaavio) porttien ja linkkien alustamisesta käyttäen GSMP-protokollaa kytkentäkentän ohjaukseen.

7.4 Konfigurointi

Kytkimen ohjausohjelmisto monine komponentteineen tarjoaa runsaasti ominaisuuksia, jotka vaativat alustusta. Ohjelmisto voidaan räätälöidä kulloisenkin tarpeen ja kokoonpanon mukaan. Kytkimen ohjausohjelmiston

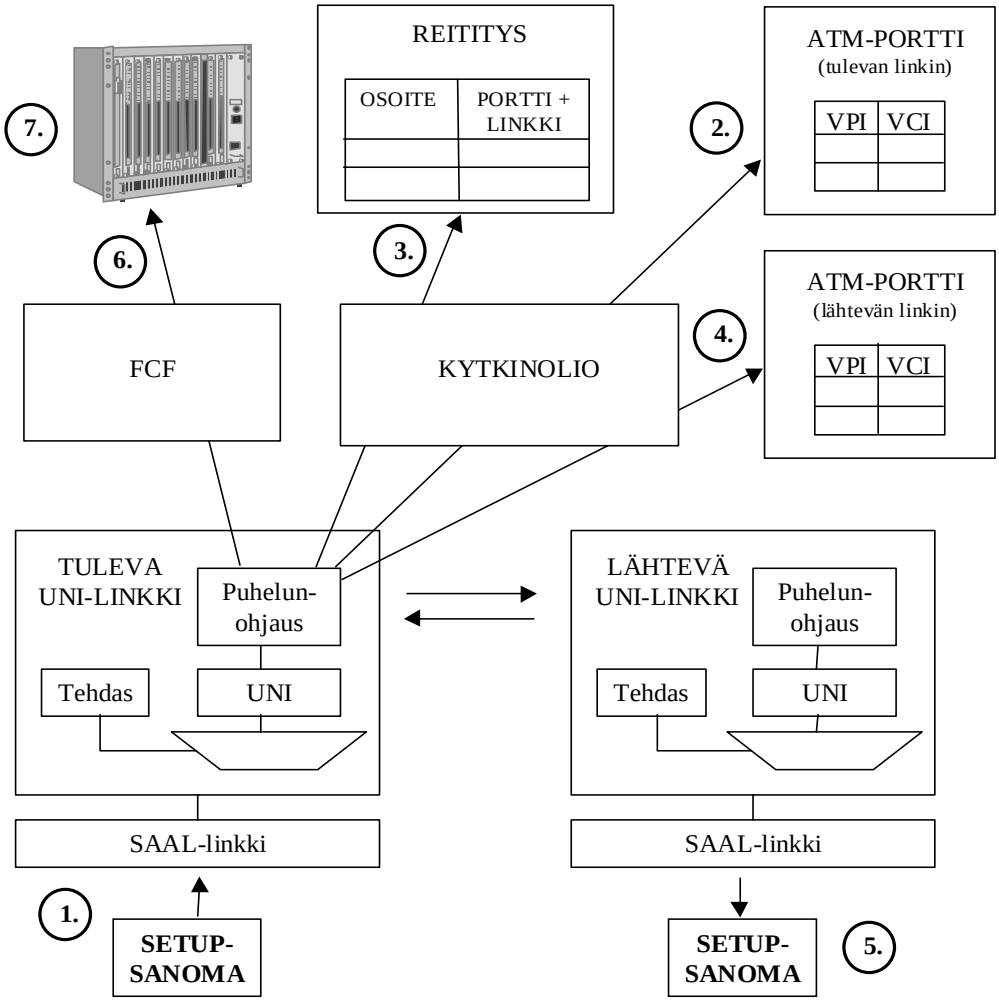
konfigurointi voi olla staattista tai dynaamista. Staattinen alustus tehdään konfiguraatiotiedostoon määriteltyjen parametrien perusteella ohjelmiston käynnistyksen yhteydessä. Dynaaminen konfigurointi on ajonaikaista ja siihen tarvitaan erillinen käyttöliittymä.

TOVE-projektissa kytkimen ohjausohjelmiston konfigurointiin suunniteltiin graafinen käyttöliittymä. Se toteutettiin projektissa kehitetyllä Java- ja komponenttipohjaisella sovelluskehyksellä (DCF, Distributed Component Framework /20/). Käyttöliittymän kautta pystyttiin lisäämään portteja ja linkkejä sekä joitakin kytkimen parametrejä, mutta suurin osa erilaisista parametreistä oli alustettu kiinteästi ohjelmistoon. Kommunikaatio käyttöliittymän ja ohjelmiston ytimen kanssa suoritettiin CORBA:lla.

SCOMS-projektissa alustettavia ominaisuuksia tuli runsaasti lisää, ja useita olemassa olevia kiinteitä parametrejä muutettiin konfiguroitaviksi. Luotiin konfiguraatiotiedosto ja sen sisällölle yksinkertainen syntaksi sekä erillinen jäsennin parametrien tulkitsemiseen. Esimerkki käytetystä konfiguraatiotiedostosta on liitteessä 3. Yksinkertainen syntaksi käy hyvin nopeasti riittämättömäksi, kun halutaan konfiguroida toisistaan riippuvia rakenteellisia parametrejä. Apuna voitaisiin käyttää olemassa olevia standardeja, kuten XML:ää (Extensible Markup Language). XML-kielen käyttö konfiguraatiotiedoston kielenä sallisi monimutkaisetkin rakenteet. Se toisi standardin tavan ja mahdollistaisi valmiiden työkalujen käytön sekä konfiguraation luomiseen että konfigurointitiedoston jäsentämiseen. Myös graafista käyttöliittymää kannattaa parantaa, jotta voidaan taata ohjelmiston konfiguroinnin dynaamisuus ja selkeys.

7.5 Puhelunmuodostus

Tässä kappaleessa käydään esimerkin avulla läpi puhelunmuodostuksen eri vaiheet ohjausohjelmiston näkökulmasta. Oheinen kuva esittää karkeasti pisteestä pisteeseen -puhelunmuodostuksen UNI-linkillä.



Kuva 28. Pisteestä pisteeseen yhteydenmuodostus UNI-linkillä.

1. Puhelunmuodostuksen merkinanto alkaa (useimmiten) käyttäjän aloittamalla yhteydenmuodostuspyynnöllä (SETUP), joka luo uuden yhteysprotokollan ja puhelunohjausinstanssin UNI-multiplekseriin kyseisessä linkkioliossa.
2. Puhelunohjaus pyytää kytkinolion kautta ko. ATM-porttia varaamaan resurssit (datakanavan VPI/VCI) tulevalle linkille.
3. Puhelunohjaus pyytää kytkinolion kautta reititysmoduulia suorittamaan reitityksen SETUP-sanoman kohdeosoitteen perusteella. Reititys-moduuli palauttaa kohdeosoitteen mukaiset portti- ja linkkinumerot.

4. Puhelunohjaus pyytää kytkinolion kautta kohteen mukaista ATM-porttia varaamaan resurssit (datakanavan VPI/VCI) lähtevälle linkille.
5. Merkinanto jatkuu vastaanottajalle ja odotetaan vastaanottajan kuittausta.
6. Vastauksen saavuttua puhelunohjaus käskee FCF:ää lähettämään kytkentäkomennon kytkinlaitteistolle.
7. Vastauksen saavuttua kytkentä on valmis. Kun suoritetaan merkinanto loppuun (eli vahvistukset käyttäjille), on pisteestä-pisteeseen puhelu kytkentäkentän läpi muodostettu.

Puhelun purkaminen alkaa vastaavalla purkupyynnöllä merkinannon keinoin. Puhelun purkamiseen kuuluvat resurssien vapautus ja kytkennän purkaminen kytkentäkentästä.

8 JOHTOPÄÄTÖKSET

Diplomityön tavoitteena oli ATM-kytkimen ohjausohjelmiston rungon toteuttaminen. Tämä sisälsi ohjelmiston ytimenä toimivan kytkinolion, portit, linkit ja kytkentäkentän ohjauksen. Kytkinolion tehtävänä oli tarjota pääsy kytkimen resursseihin sekä mahdollistaa ohjelmiston eri osien yhteistoiminta. Ohjelmiston tuli olla helposti laajennettavissa monikäyttö-kytkimen tarpeisiin.

Työssä saatiin aikaan toimiva ATM-kytkimen ohjausohjelmiston ydin, joka hallitsee ohjelmiston muiden moduulien toimintaa. Ohjausohjelmistoon pystytään helposti vaihtamaan halutut reititys- ja kytkentäkentänohjausoliot sekä lisäämään verraten helposti uuden tyyppisiä portteja ja linkkejä.

Diplomityön kirjallinen osuus kuuluu osana SCOMS-projektin dokumentaatiota. Siinä kuvataan ohjausohjelmiston osia ATM-tekniikan näkökulmasta, sekä ohjelmiston toteutusta ja rakennetta. Myös toteutuksen aikana esille tulleita ideoita esitellään mahdollisen jatkokehityksen pohjaksi. Työssä tutkittiin oliomenetelmien käyttöä ohjausohjelmiston toteutuksessa. Käytettyjä suunnittelun ja toteutuksen apuvälineitä olivat suunnittelumallit, UML, CORBA ja OVOPS++-sovelluskehys.

Oliomenetelmät tuovat ohjelmistokehitykseen yhä enemmän koodin uudelleenkäyttöä ja modulaarisuutta. Ennen kaikkea olio-ohjelmointikielten tarjoama luokkahierarkia yhdistettynä suunnittelumalleihin mahdollistaa helposti laajennettavan ohjelmiston toteuttamisen. Tämä vaatii kuitenkin ohjelmisto-suunnittelijalta vankkaa kokemusta oliomenetelmistä. Olio-ajattelun sisäistäminen vie aina oman aikansa, ja usein kokemattoman ohjelmoitsijan koodi onkin ohjelmiston jatkokehityksen kannalta huonosti hyödynnettävissä.

LÄHTEET

/1/ M. Grundström, R. Mickos, ATM-tekniikka ja monipalveluverkot, 2. painos, Suomen Atk-kustannus Oy, Espoo, 1998, ISBN 951-762-488-3.

/2/ H.J.R. Dutton, P. Lenhard, Asynchronous Transfer Mode (ATM) Technical Overview, Second Edition, Prentice-Hall, 1995, ISBN 0-13-52044-5.

/3/ R.A.M. Sprenkels, Management of ATM Networks, Master of Science Thesis, Enschede, June 1996.
http://wwwhomecs.utwente.nl/~sprenkel/thesis

/4/ ITU-T, Recommendation E.164 – Numbering Plan for the ISDN Era, Geneva, 1991.

/5/ Private Network-Network Interface Specification, Version 1.0, The ATM Forum Technical Committee, af-pnni-0055.00, March 1996

/6/ The Next-Generation ATM Switch: From Testbeds to Production Networks, Cisco Systems, 1996.

/7/ A. Alles, ATM Interworking, Cisco Systems, May 1995.

/8/ W. Stallings, SNMP, SNMPv2, and CMIP, The Practical Guide to Network-Management Standards, Addison-Wesley, 1993, ISBN 0-201-63331-0.

/9/ Integrated Local Management Interface (ILMI) Specification, Version 4.0, The ATM Forum Technical Committee, af-ilmi-0065.00, Septemper, 1996.

/10/ P. Newman, W. Edwards, R. Hinden, E. Hoffman, F. Ching Liaw, T. Lyon, G. Minshall, Ipsilon's General Switch Management Protocol Specification Version 2.0 (RFC 2297), IETF, 1998.

/11/ T. Haatanen, J. Tuliluoto, FSR switch Application Programming Interface User's Guide, Version: 2.0, VTT, May 1998

/12/ M. Fowler, K. Scott, UML Distilled: Applying the Standard Object Modeling Language, Addison-Wesley, 1998, ISBN 0-201-32563-1.

/13/ OMG Unified Modeling Language Specification (draft), version 1.3 alpha R5, Object Management Group Inc., March 1999.

/14/ E. Gamma, R. Helm, R. Johnson, J. Vlissides, Design Patterns: Elements of Reusable Object-Oriented Software, Addison-Wesley, 1995, ISBN 0-201-63361-2.

/15/ S. Vinoski, CORBA: Integrating Diverse Applications Within Distributed Heterogeneous Environments, IEEE Communications Magazine, Vol. 35, No. 2, February 1997.

/16/ H. Hüni, R. Johnson, R. Engel, A Framework for Network Protocol Software, OOPSLA'95 Proceedings, Austin, 1995.

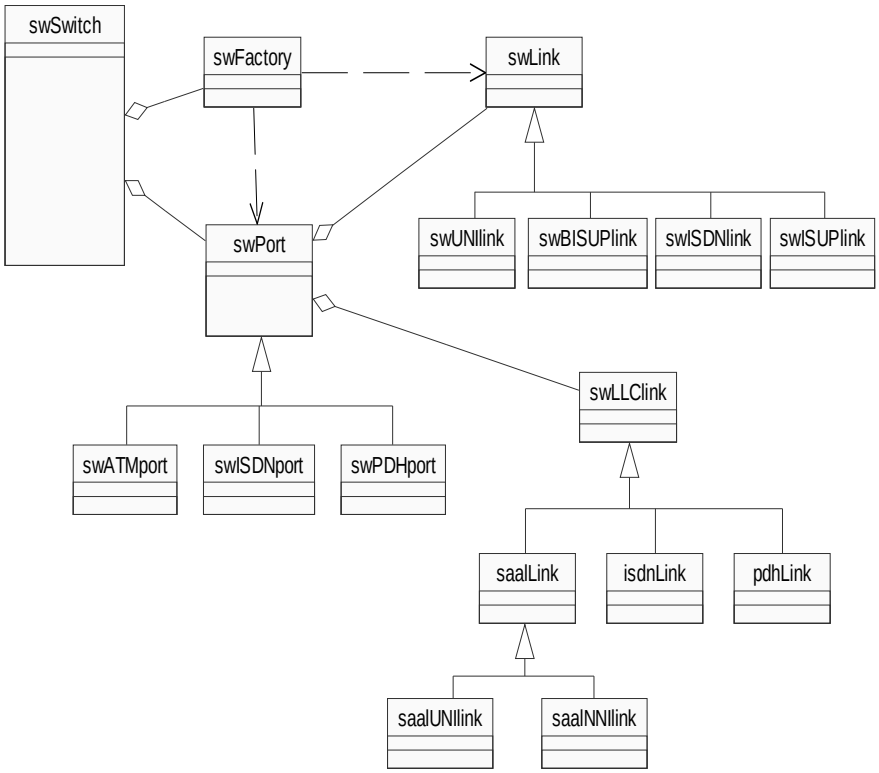
/17/ O. Martikainen, V-M. Puro, J. Räsänen, T. Pärnänen, P. Nummisalo, P. Koponen, TOVE-projecet 1996 Deliverables, 1996.

/18/ O. Martikainen, P. Puro, J. Sonninen, OVOPS, Object Virtual Operations System for Distributed Applications Development.

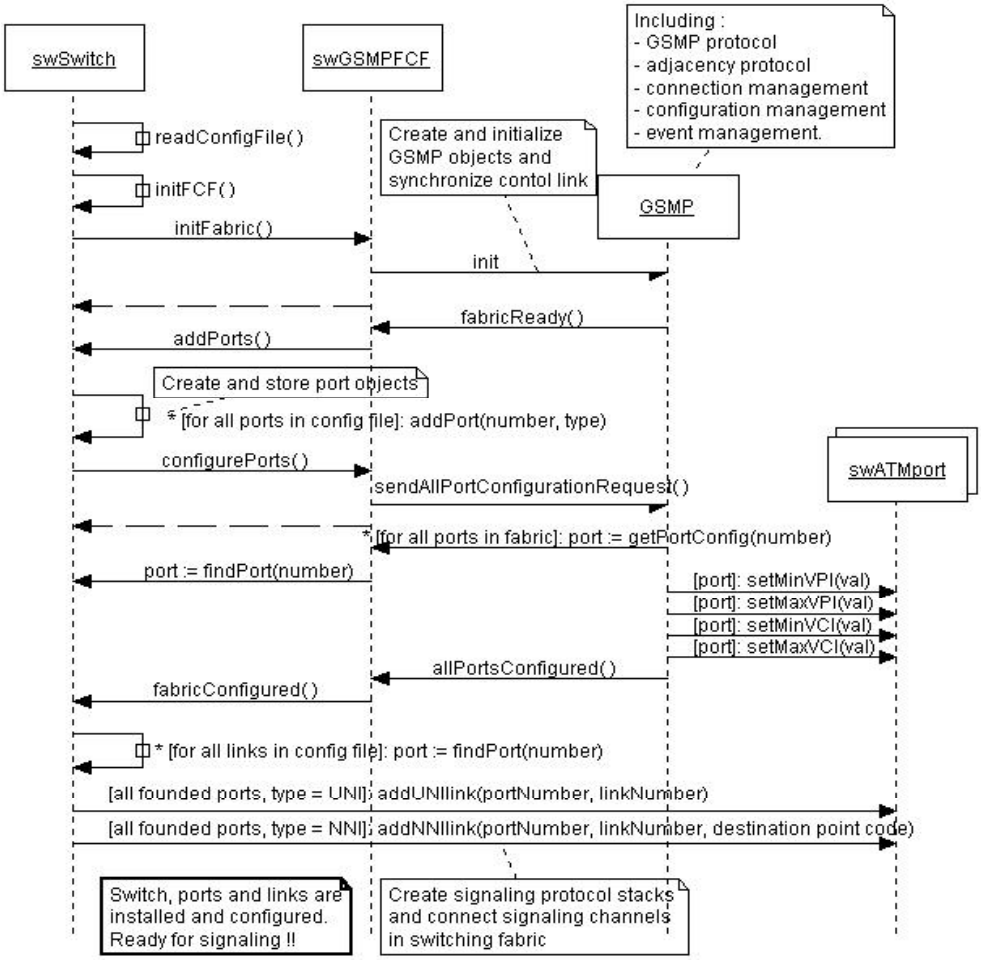
/19/ G. Gullekson, B. Selic, Design Patterns for Real-Time Software, Embedded Systems Conference, West ’96.

/20/ P. Nummisalo, Uudet olioteknologiat älyverkkototeutuksessa. Diplomityö, Lappeenrannan teknillinen korkeakoulu, Tietotekniikan osasto, 1999.

LIITE 1. Luokkakaavio porttien ja linkkien rakenteesta.



LIITE 2. Porttien ja linkkien alustus GSMP:llä.



LIITE 3. SCOMS-kytkimen konfiguraatiotiedosto.

```
#
# Switch configuration file
#

# Control port indicates controller's (PC) control port (default 0)
CONTROL_PORT 0

# Fabric control port indicates fabric (switch hardware) control port
# where controller is connected to (default 1)
FABRIC_CONTROL_PORT 2

# Signaling VPI/VCI values (default 0/5)
SIGNALING_VPI 0
SIGNALING_VCI 5

# Signaling VPI/VCI values (default 0/16)
ILMI_VPI 0
ILMI_VCI 16

# Signaling VPI/VCI values (default 0/15)
GSMP_VPI 0
GSMP_VCI 86

# Minimum and maximun values for VPI/VCI values of data channel.
# This VCI range can be restricted with set methods in sw module
# (e.g called by switching fabric), but not extended !!
# Only VPI = 0 is supported in this version, following VPI values
# have no effect (default VCI range 32-65535).
MIN_VPI 0
MAX_VPI 0
MIN_VCI 100
MAX_VCI 65535

# Base values for special VCI reservation
# (default base for signaling vci is 40 and for ILMI vci 60)
SIGNALING_VCI_BASE 40
ILMI_VCI_BASE 60

# Select used modules (default values for all is 'used' (1))
ILMI_USED 0
SS7_USED 0

# Select fabric control function (GSMP, API, NOT_USED)
# (default value is NOT_USED)
FABRIC GSMP

# Routing (SIMPLE, TRS)
# Default routing is SIMPLE routing.
ROUTING SIMPLE

# MANDATORY config variable POINT_CODE (no default value available)
# This SS7 point code is used also for CORBA Nameservice to identify
# a switch instance
POINT_CODE 1

# Network prefix used in address registration
# (no default value set (empty string))
NET_PREFIX 47.0005.80FFE1000000F20F4A15
```

```
# Debug trace NULL, COUT, FILE, CORBA (default is off (NULL))
# Note: Only COUT is noticed, other values are interpreted like NULL
# Debug levels are 1,2,3 (default is most quiet (1))
DEBUG_TRACE COUT
DEBUG_LEVEL 3

# Ports to be installed. Lists port numbers (uses FSR155 port type).
# NOTE: If FABRIC is NOT_USED, this variable has no effect, because
#       switching hardware installs ports automatically.

BEGIN PORTS
# type port numbers
  ATM 2 3 4
#  PDH 2 3
#  ISDN 5 6
END

# Links to be installed. List link type, port number and link number.
# NNI links need also point codes.
# NOTE: If corresponding ports are not installed, exception handling
#       prevents to install links.

BEGIN LINKS
# port link type [point code]
# ATM (UNI31, UNI40, DSS2, BISUP)
  2 1 UNI31
  3 1 UNI31
  4 1 UNI31

#  2 1 BISUP 2
#  3 1 BISUP 3
# PDH
#  2 1 ISUP 2
#  3 1 ISUP 3
# ISDN
#  5 1 DSS1
#  6 1 DSS1
END

BEGIN SS7_USERS
  TCAP
  BISUP
  ISUP
END

BEGIN MTP3_ROUTES
# start point code, end point code, port number, link number
  2 2 2 1
  3 3 3 1
END

# File name for manually configured (without ILMI) addresses
# NOTE: If ILMI_USED is set (1), this variable has no effect, because
#       ILMI registers addresses automatically.
ADDRESS_FILE_NAME address.cfg
```