LAPPEENRANNAN TEKNILLINEN KORKEAKOULU 4.5.1999
Tietotekniikan osasto
Tietoliikennetekniikan laitos

1715 TIETOTEKNIIKAN ERIKOISTYÖT

Loppuraportti

B-ISUP-protokollan toteutus

Tarkastaja: prof. Jorma Jormakka
Tekijä: 5314 Sami Raatikainen, Tite 5

SISÄLLYSLUETTELO

LYHENTEET
1 JOHDANTO 4
2 B-ISUP 6
2.1 Johdanto 6
2.2 B-ISDN SS7 6
2.3 Standardit 8
2.4 Arkkitehtuuri 8
2.5 Toteutettavat ominaisuudet 11
3 OVOPS++ 12
3.1 Johdanto 12
3.2 Arkkitehtuuri 13
3.3 Tärkeimmät komponentit 14
4 TOTEUTUS 16
4.1 Lähtökohta 16
4.2 Toteutusvaihtoehto 1 17
4.3 Toteutusvaihtoehto 2 18
4.4 Toteutusvaihtoehto 3 19
4.5 Valinta ja perustelut 20
5 TOTEUTUSYKSITYISKOHTIA 22
5.1 Johdanto 22
5.2 B-ISUP AE 22
5.3 NI 23
5.4 Primitiivit 24
6 TESTAUS JA JATKOKEHITYS 25
7 JOHTOPÄÄTÖKSET 26
LIITE 1. Toteutetut tilat.

LYHENTEET

AE	Application Entity
AINI	ATM Inter-Network Interface
ASE	Application Service Element
ATM	Asynchronous Transfer Mode
BCC	Bearer Connection Control
B-ICI	Broadband Inter Carrier Interface
B-ISDN	Broadband ISDN
B-ISUP	Broadband ISDN User Part
CC	Call Control
CORBA	Common Object Request Broker Architecture
CS-1	Capability Set 1.
FSR	Frame Synchronized Ring
GSMP	General Switch Management Protocol
IAM	Initial Address Message
IISP	Interim Interswitch Signalling Protocol
ISDN	Integrated Services Digital Network
ISUP	ISDN User Part
ITU	International Telecommunications Union
MC	Maintenance Control
OSI	Open Systems Interconnect
OVOPS	Object-oriented Virtual Operations System
PDU	Protocol Data Unit
PF	Protocol Framework
PNNI	Private Network-to-Network Interface
QoS	Quality of Service
SAAL	Signalling ATM Adaptation Layer
SACF	Single Association Control Functions
SCCP	Signalling Connection Control Part
SDL	Specification and Description Language
SF	Scheduling Framework

SID	Signalling Identifier
SIO	Service Information Octet
SS7	Signalling System no.7
SSCF	Service Specific Coordination Function
SSCOP	Service Specific Connection-Oriented Protocol
N-ISDN	Narrowband ISDN
NI	Network Interface
NNI	Network-to-Network Interface
TCAP	Transaction Capabilities Application Part
TEKES	Tekniikan edistämiskeskus
TKK	Teknillinen korkeakoulu
TOVE	Toimialaverkot-projekti
UI	Unrecognized Information
UNI	User-to-Network Interface
VTT	Valtion Teknillinen Tutkimuskeskus

1 JOHDANTO

B-ISUP-protokolla (Broadband ISDN User Part), jonka suunnittelua ja toteutusta tässä erikoistyössä kuvataan, on tehty Teknillisen korkeakoulun (TKK) tietoliikenneohjelmistojen ja multimedian laboratoriolle osana TOVE-projektia (Toimialaverkot). TOVE on vuonna 1996 alkanut TEKES:n (Tekniikan edistämiskeskus) ja yhteistyöyritysten rahoittama kolmivuotinen tutkimushanke, jonka tavoitteeksi oli asetettu laajakaistaisen ISDN-kytkimen (engl. Broadband Integrated Services Digital Network, B-ISDN) ohjelmiston suunnittelu ja toteutus. Ohjelmisto koostuu OVOPS++-protokollasovelluskehyksestä (Object-oriented Virtual Operations System) ja sen avulla toteutettavista merkinantoprotokollista, kytkimen ohjausosasta ja älyverkko-osasta. /1/

Fyysisenä laitteena projektissa toimii VTT:n (Valtion teknillinen tutkimuskeskus) kehittämä FSR-kytkentäarkkitehtuuriin (Frame Synchronized Ring) pohjautuva ATM-kytkin. Ohjelmisto sisältää myös tuen GSMP-protokollalle (General Switch Management Protocol), jolla tätä kytkintä ohjataan. /2/

B-ISUP-protokolla toteuttaa projektissa ja kytkimessä NNI-rajapinnan (Network-to-Network Interface) yhteiskanavamerkinannon, eli merkinannon eri verkkoelementtien kesken. Se, kuten suurin osa muistakin projektissa toteutettavista protokollista, pohjautuu ITU-T:n (International Telecommunications Union standardization sector) standardeihin. Muita projektissa toteutettuja ohjelmiston osia tai protokollia ovat mm. käyttäjäpuolen merkinannot Q.2931 ja UNI3.1 (ATM Forum), CS-1-mallin (Capability Set 1) mukainen puhelunohjausmalli, SAAL-protokollat SSCOP, UNI-SSCF ja NNI-SSCF, SS7-protokollat TCAP, SCCP ja MTP-3, ja CORBA (Common Object Request Broker Architecture) Intelligent Network Gateway.

Tämän työn tavoitteena on löytää sopiva toteustapa laajakaistaverkon (B-ISDN) yhteiskanavamerkinantojärjestelmän ISDN-käyttäjäosan B-ISUP-protokollan prototyypin toteuttamiseen oliopohjaisella OVOPS++-protokollasovelluskehyksellä, sekä kertoa valitusta ratkaisusta ja siihen liittyvistä asioista. Aivan aluksi työssä kuitenkin käydään läpi taustaa protokollan valintaan projektissa, sekä tutustutaan sen teoriaan. Suunnittelussa kerrotaan mistä lähdettiin liikkeelle, mihin standardeihin toteutus pohjautuu, sekä miten toteutusta suunniteltiin.

Toteutuksen suunnittelussa tärkeänä osana on myös käytetyn protokollasovelluskehyksen (OVOPS++) ominaisuuksien tarkastelu ja huomiointi. Ja ennen toteutustapojen esittelyä ja vertailua käydäänkin läpi sovelluskehyksen tärkeimmät komponentit ja ominaisuudet. Tässä työssä tutkitaan erilaisia toteutustapoja, eikä keskitytä suoraan yhteen tapaan. Voidaankin sanoa, että tässä työssä keskitytään protokollan teoriaan, toteutuksen suunnitteluun ja varsinaiseen toteutukseen. Valituksi tulleen toteutustavan hahmotuttua selvitetään miksi päädyttiin juuri siihen ratkaisuun. Valitusta toteutustavasta kerrotaan hieman tarkemmin, sekä muutamiin yksityiskohtiin pureudutaan syvemmin. Alustava aikataulu esiteltiin alkuraportissa, sekä väliraportissa selvitettiin tilanne sillä hetkellä. Lopuksi tässä raportissa katsotaan, kuinka aikataulun suunnittelu oli onnistunut kokonaisuudessaan, sekä vedetään johtopäätökset toteutuksesta ja koko suunnitteluprosessista.

2 B-ISUP

2.1 Johdanto

B-ISUP valittiin TOVE-projektin alkuperäisissä suunnitelmissa merkinannoksi kytkimen NNI-rajapintaan. Sen valintaan ei ollut olemassa mitään tarkemmin määriteltyjä syitä, jokin protokolla täytyi vain valita verkkorajapintaan. Yksi valintaan vaikuttavia syitä kuitenkin oli mm. ITU-T:n määrittelemän Q.2931-protokollan valinta käyttäjärajapinnan merkinannoksi, ja B-ISUP olisi siten luonnollinen valinta verkkorajapintaan. Lisäksi projektissa muutenkin noudatettiin ITU-T:n standardeja.

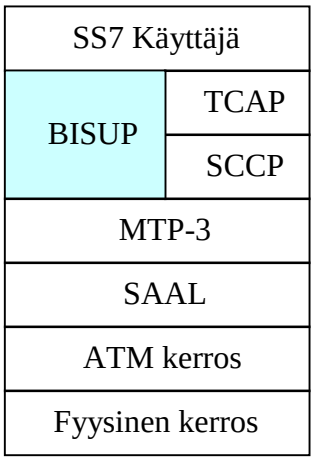
Muita mahdollisia NNI-rajapinnan merkinantoprotokollia olisivat mm. PNNI (Private Network-to-Network Interface), B-ICI (Broadband Inter Carrier Interface) ja IISP (Interim Interswitch Signalling Protocol), jotka kaikki ovat ATM Forumin määrittelemiä ratkaisuja. Niistä PNNI on tarkoitettu vain yksityisten verkkojen merkinannoksi ja B-ICI vain julkisten verkkojen (operaattorien) merkinannoksi. Nyt on nousemassa uutena mielenkiintoa herättävänä toteutuksena myös AINI (ATM Inter-Network Interface), mikä on ATM Forumin määrittelemä uusin rajapinta ATM-verkkojen kytkemiseen. Välttääkseen yhteensopivuusongelmat ja mahdollistaakseen nopean kehityksen ja toteutuksen, AINI on määritelty tukemaan vain sellaisia ominaisuuksia jotka ovat määriteltyjä sekä PNNI- että B-ISUP-protokollissa. /3/

2.2 B-ISDN SS7

Merkinantojärjestelmät voidaan jakaa niiden teknisen toteutuksen perusteella kanavakohtaiseen merkinantoon ja yhteiskanavamerkinantoon. Kanavakohtaisissa jokaisella tiedonsiirtoyhteydellä on oma merkinantokanavansa. Yhteiskanavamerkinantojärjestelmä taas käyttää yhteistä merkinantokanavaa useammalle tiedonsiirtoyhteydelle.

SS7-merkinantoverkko (engl. Signalling System no.7, SS7) on yhteiskanavamerkinantojärjestelmä ja käytössä digitaalisissa televerkoissa sisältäen toiminnot liikenteen ohjaukseen ja hallintaan. Nykyisin sitä käytetään ISDN-verkkojen lisäksi mm. GSM- ja IN-verkoissa. SS7-merkinantoverkko sisältää sanomansiirto-osan (engl. Message Transfer Part, MTP), joka tarjoaa tarjoaa luotettavan ja yhteydettömän siirtojärjestelmän merkinantopisteiden välillä. Tämä sanomansiirto-osa on määritelty kolmeen toiminnalliseen tasoon sekä hallintaosaan. Merkinantolinkki (MTP-1) vastaa fyysisen tason toiminnoista, merkinantokanava (MTP-2) linkkikerroksen toiminnoista ja merkinantoverkko (MTP-3) lähinnä verkkokerroksen toiminnoista OSI-viitemalliin verratessa. MTP:n käyttäjäosa on ISDN-verkoissa ISUP (ISDN User Part). Lisäksi järjestelmään kuuluvat merkinantoyhteyden ohjausosa (engl. Signalling Connection Control Part, SCCP) ja tapahtumankäsittelyosa (engl. Transaction Capabilities Application Part, TCAP).

Laajakaistainen SS7 on johdettu hyvin pitkälti perinteisemmästä kapeakaistaisesta SS7-mallista. Samoin B-ISUP on määritelty ISUP:n pohjalta. B-ISUP sisältää merkinannon toiminnot yhteyksien hallintaan ja palvelujen tason käsittelyyn B-ISDN-verkoissa. B-ISUP käyttää hyväkseen MTP-3-protokollan tarjoamaa palvelua täysin vastaavasti ISUP:n kanssa. MTP-3 omaa hyvin samanlaiset toiminnot riippumatta siitä, onko kyseessä N-ISDN- vai B-ISDN-verkko. B-ISUP muodosti tavallaan käsitteen uusi ISUP (engl. new ISUP), mikä näkyy mm. uutena SIO-arvona (Service Information Octet) MTP-3-kerrokselle. /5/ SIO-arvot määrittävät MTP-tason käyttäjät, joiden avulla MTP-3 ohjaa sanomat oikeisiin pisteisiin. Alemmat MTP-tasot voidaan korvata B-ISDN-verkoissa SAAL- (Signalling ATM Adaptation Layer) ja ATM-kerroksilla, sekä fyysisellä kerroksella.



Kuva 1. B-ISDN SS7 protokollapino.

2.3 Standardit

Tässä työssä tutkittava, suunniteltava ja toteutettava B-ISUP pohjautuu ITU-T:n standardisarjoista Q-sarjaan, johon kuuluvat telealan kytkentään ja merkinantoon liittyvät standardit (Series Q Recommendations – Switching and Signalling /4/). Tarkoituksena on toteuttaa B-ISUP-protokollan prototyyppi ITU-T:n standardien Q.2761-Q.2764 mukaan puhelunohjauksen (Call Controller, CC) osalta. Nämä standardit toteuttavat sen CS-1-mallin mukaisen peruspuheluprosessin yhteyden luomisen, ylläpitämisen ja purkamisen. Q.2761 määrittelee protokollan yleiskuvauksen, Q.2762 viestien ja signaalien tarkoitukset, Q.2763 viestien sisällön ja Q.2764 peruspuheluprosessin tarkemman kuvauksen. /5/ /6/ /7/ /8/

2.4 Arkkitehtuuri

B-ISUP:n toiminnallisuus jakaantuu kahteen suurempaa osaan, protokollakohtaisiin ja ei-protokollakohtaisiin toimintoihin. Ei-protokollakohtaiset toiminnot kuvaavat käyttäjän, eli sovelluksen (engl. Application Process, AP), toimintoja protokollan yläpuolella. Näitä ovat CS-1-standardin mukainen puhelunohjaus (engl. Call Control, CC), hallintatoiminnot (engl. maintenance functions) ja yhteensopivuustoiminnot

(engl. compatibility functions). Näistä TOVE-toteutus sisältää vain CC:n toiminnot, jotka ovat erillisessä jo aiemmin projektissa toteutetussa protokollassa. Hallintatoiminnot ja yhteensopivuustoiminnot eivät ole välttämätön osa. Mutta niiden toimintoja on sisällytetty osaksi B-ISUP- ja CC- protokollien toteutusta. Protokollakohtaiset toiminnot jakaantuvat viiteen pienempään toiminnalliseen lohkoon, sovelluspalveluelementteihin (engl. Application Service Element, ASE).

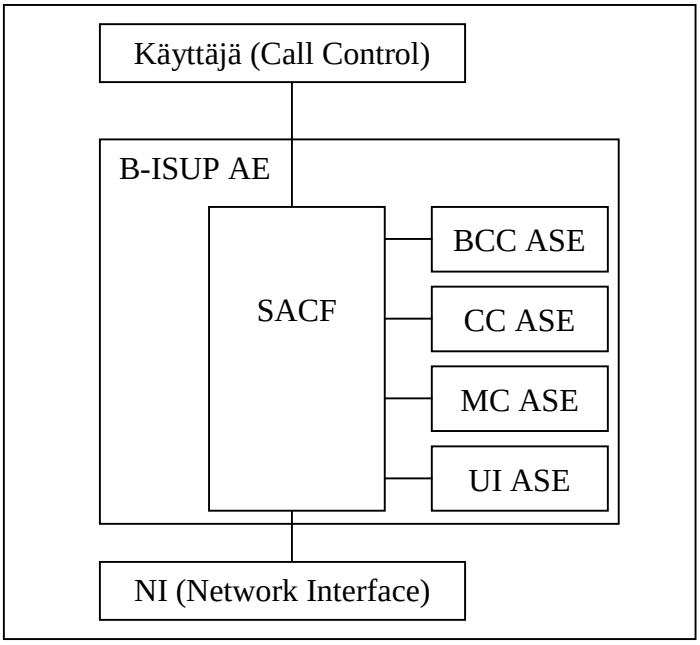
BCC ASE (Bearer Connection Control) vastaa yhteyden muodostamisesta ja purkamisesta vastapuolen merkinantopisteen kanssa. CC ASE (Call Control) huolehtii pääasiassa viestien sekvenssin tarkistuksesta, sekä primitiivien muuntamisesta ylärajapinnan ja B-ISUP:n sisäisten signaalien kesken. Molemmat näistä sisältävät myös ajastimia virhetilanteiden havaitsemiseksi.

MC ASE (Maintenance Control) toteuttaa joitakin hallintatoimintoja kuten resurssien blokkaus ja palautus. Resurssin blokkaus voisi aiheutua vaikka väliaikaisesti katkenneesta linkistä. UI ASE (Unrecognized Information) määrittelee käsittelyn tunnistamattomille viesteille. Tämä toiminto on kuvattu hyvin standardinomaisesti, eikä varsinaisesti määrittele mitään selkeätä toimintoa. UI ASE määritellään vain pääasiassa lähettämään tunnistamattomat viestit yläpuoliselle yhteensopivuustoiminnoista vastaavalle sovellukselle, joka sitten päättäisi lopullisen toiminnan.

Näitä neljää sovelluspalveluelementtiä hallitsee yksittäisen sidoksen kontrollifunktio (engl. Single Association Control Function, SACF). Se on vastuussa sekä sovellukselta että alemmalta kerrokselta tulevien primitiivien ohjaamisesta ja hajauttamisesta sisäisiksi signaaleiksi oikeille sovelluspalveluelementeille. Lisäksi se vastaa sovelluspalveluelementeiltä tulevien signaalien kokoamisesta ja ohjaamisesta joko sovellukselle tai alemmalle kerrokselle. Eli SACF ohjaa, kokoaa, hajauttaa ja synkronisoi primitiivejä ja sisäisiä signaaleja B-ISUP-sovelluksessa.

SACF, BCC ASE, CC ASE, MC ASE ja UI ASE määrittelevät B-ISUP-sovelluskokonaisuuden (engl. Application Entity, AE). Jokaista merkinantoyhteyttä kuvaa yksi tällainen sovelluskokonaisuus. Standardi Q.2764 määrittelee käsitteelliset rajapinnat näiden kaikkien lohkojen välille, mutta nämä rajapinnat eivät ole todellisia tai testattavia rajapintoja. Käytännössä se tarkoittaa sitä, että toteutustapa on vapaa ja sisäinen toiminto sovelluskokonaisuudessa voidaan toteuttaa omalla tavalla.

Standardissa Q.2764 määritellään myös rajapintatoiminnot NI (Network Interface), SACF:n ja MTP-3:n väliin. Yksi NI toimii kaikkien siinä linkissä olevien B-ISUP AE-instanssien (merkinantoyhteys) ja yhden MTP-3-protokollan välissä. NI lähettää B-ISUP:n viestit vastinoliolle standardin Q.2763 määrittäminä bittikehyksinä käyttäen MTP-3-protokollan tarjoamaa kuljetuspalvelua. MTP-3:lta tulleet viestit NI ohjaa oikeaan B-ISUP-sovellukseen. Jos B-ISUP-sovellusta kyseiselle yhteydelle ei ole vielä olemassa, NI:n täytyy osata luoda sellainen. NI suorittaa lisäksi joitain yksinkertaisia virheentarkistuksia viesteistä, kuten pituusrajat.

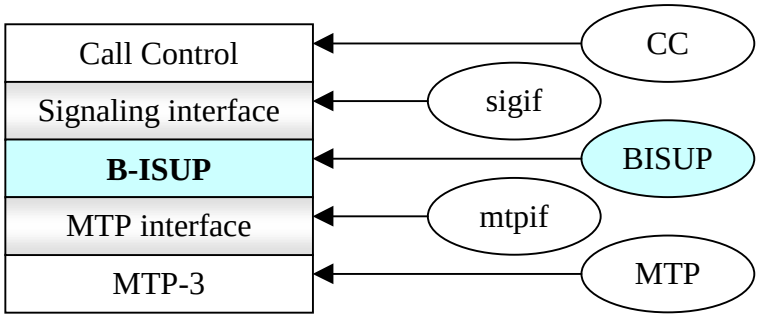


Kuva 2. B-ISUP arkkitehtuuri.

2.5 Toteutettavat ominaisuudet

Alkuperäinen TOVE-projektin projektisuunnitelma määritteli B-ISUP:n toteutuksen kattavan CS-1-mallin mukaisen prototyypin, joka toteuttaa CC ja BCC ASE:n toiminnot kokonaisuudessaan. MC ASE:a ei toteuteta, koska vastaavia hallintatoimintoja ei ollut toteutettu TOVE:n MTP-3-protokollaan, mutta näiden toimintojenkin toteutus tulisi olla mahdollista lisätä jossain vaiheessa. UI ASE:n toiminnot on määritelty standardissa Q.2764 hyvin abstraktilla tasolla, eikä varsinaista konkreettista toimintaa ole ollenkaan kuvattu. Siksi tätä ei toteuteta sellaisenaan, mutta sen toimintaa pyritään sisällyttämään kokonaisuuteen mahdollisimman hyvin. Eli käytännössä määritellään toiminnot tunnistamattomille viesteille.

TOVE-projektissa ohjelmisto muodostuu erikseen toteutettavista moduleista (hakemisto). Yleisesti yksi protokolla tai jonkin protokollan tarjoama rajapinta vastaa yhtä modulia. MTP-3-moduli oli jo toteutettu ennen B-ISUP-toteutuksen alkamista SCCP:tä ja IN:ää varten, sekä ennakoiden B-ISUP:n toteutusta. Lisäksi MTP:n tarjoama primitiivirajapinta oli toteutettu omaan moduliin, joka sisälsi kaikki MTP-3:n määrittelemät primitiiviluokat. Lisäksi projektissa oli jo CC-moduli, joka toteuttaa CS-1-standardin mukaisen peruspuhelumallin (engl. Basic Call State Model, BCSM). Primitiivirajapinta CC:n alla oli myös määritelty ja toteutettu aiemmin projektissa UNI3.1- ja Q.2931-standardien mukaiseksi. Tämä rajapinta oli tarkoitus hyödyntää myös B-ISUP:n toteutuksessa.



Kuva 3. B-ISUP toteutukseen liittyvät TOVE-modulit.

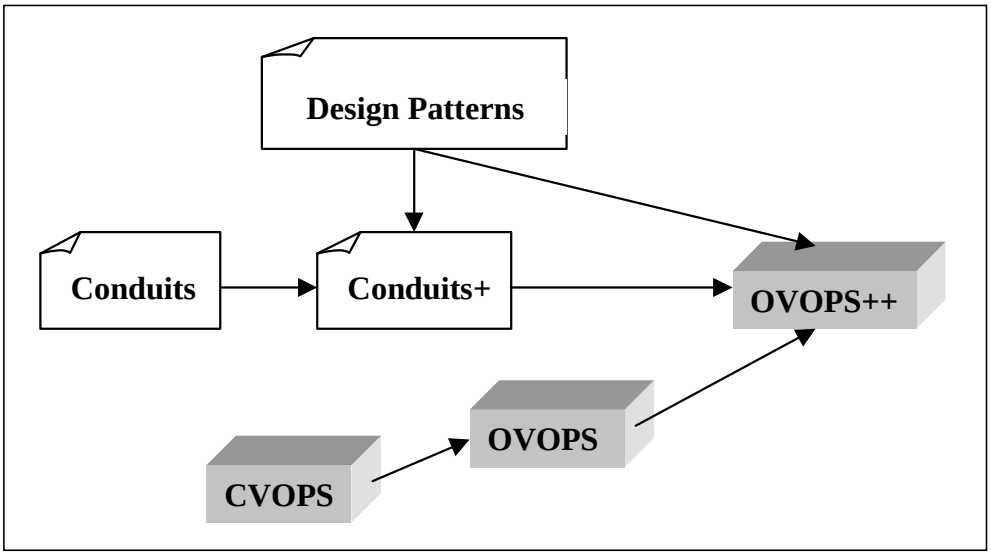
3 OVOPS++

3.1 Johdanto

Sovelluskehyksiä luodaan ja kehitetään helpottamaan ja nopeuttamaan ohjelmistokehittäjien työtä. Sovelluskehys (framework) tarjoaa sovellusaluekohtaisen rungon (olio-ohjelmoinnissa joukon luokkia ja luokkahierarkkioita sekä oletustoiminnan kuvaavan toimintalogiikan), jota täydentämällä ja tarkentamalla luodaan kulloinen toteutettava sovellus. TOVE-projektissa on ollut käytössä sekä kehitettävänä OVOPS++-protokollasovelluskehysympäristö.

OVOPS++ perustuu Ascom Tech AG:n, University of Illinoisin ja GLUE Software Engineeringin julkaisemien papereiden pohjalta kehitettyyn Conduits+ -sovelluskehykseen. /9/ Se on oliosuuntautunut sovelluskehysmalli tietoliikenneohjelmistojen toteuttamiseen, ja tarjoaa malleja protokollien ja niiden välisten yhteyksien toteuttamiseen. Se käyttää tehokkaasti hyväkseen komponentteihin perustuvaa oliosuuntautunutta sovelluskehysajatusta, sekä oliosuuntautuneita suunnittelumalleja (engl. design patterns /10/). Conduits+ -sovelluskehystä on käytetty muun muassa ATM-kytkimen toteuttamiseen, joten se oli luonnollinen valinta myös OVOPS++ -ympäristöön. Conduits+ -sovelluskehys integroitiin Telen rahoituksella kehitettyyn OVOPS-ohjelmistoon (Object Virtual OPerations System), joka toteuttaa protokollien vaatiman ajonaikaisen ympäristön sisältäen luokkia esimerkiksi ajan jakoa, ajastimia ja I/O-ajureita varten. Sovelluskehyksen nimi OVOPS++ tulee siis OVOPS:n ja Conduits+:n yhdistämisestä, ja toinen plus tarkoittaa edistyneempää Conduits+:a. Ohjelmisto on tehty C++ -kielellä ja se toimii tällä hetkellä Linux-käyttöjärjestelmässä, mutta sen porttaaminen johonkin toiseen ympäristöön, joka tarjoaa riittävän modernin C++ -kääntäjän, olisi myös mahdollista.

Kaikki TOVE-projektissa kehitetyt protokollat on tehty OVOPS++ -ympäristöön. Ympäristöä on kehitetty projektin edetessä rinnan varsinaisen protokollakehityksen kanssa. Sovelluskehysten kehittäminen, kuten olio-ohjelmointi yleisestikin, onkin luonteeltaan iteratiivinen prosessi.



Kuva 4. OVOPS++-sovelluskehyksen kehitys.

3.2 Arkkitehtuuri

OVOPS++ muodostuu hyvin määritellyistä uudelleen käytettävistä rakennuspalikoista joita käyttäen protokolla voidaan koota. Näin toteutuksesta saadaan usein hyvinkin suoraviivainen prosessi ja koodin uudelleenkäytöstä tehokasta. OVOPS++:a käytettäessä sovelluskehyksen luokista peritään protokollakohtaiset luokat, joihin toiminnallisuus täydennetään.

OVOPS++ tarjoaa tällä hetkellä tavallaan kaksi erillistä sovelluskehystä, protocol framework (pf) ja scheduling framework (sf), jotka toimivat kuitenkin tiiviisti toistensa ehdoilla. Pf:n tarjoaa eri komponenttien sisäisen toiminnan ja rajapinnan (julkinen ja perittävä), sekä rajapinnat joidenkin komponenttien yhteenliittämiseen ja niiden väliseen kommunikointiin. Sf on

vastuussa ajan jaosta prosessissa ja enemmän piilossa suunnittelijalta. Sf:n käyttö näkyykin lähinnä vain pf:n tarjoamien kantaluokkien sisällä, eli pf:n komponentit pyytävät suoritusaikaa sf:ltä.

OVOPS++-arkkitehtuurin pf-osa voidaan jakaa karkeasti kahteen suurempaan kategoriaan, tietoa käsitteleviin osiin (conduits) ja tietoa esittäviin osiin (information chunks). Conduit on tietokanava, jolla on kaksi puolta (A- ja B-puoli) ja jonka avulla voidaan kuljettaa sopivaa tietoa toiselta puolelta toiselle puolelle ja eri conduitien välillä.

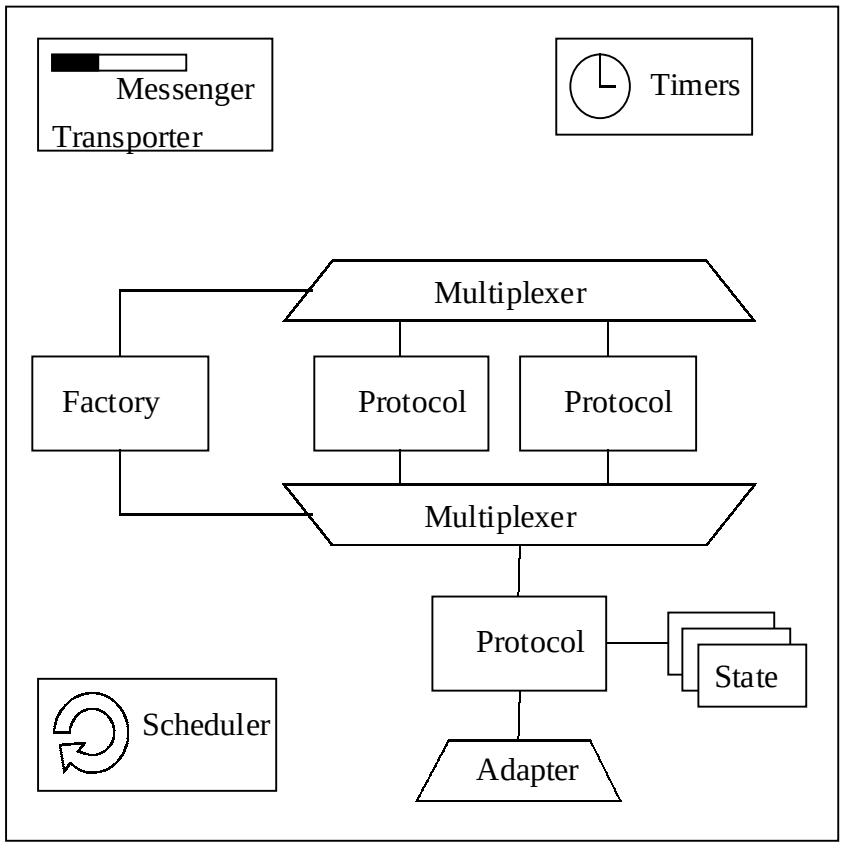
3.3 Tärkeimmät komponentit

Merkittävimmät tietoa käsittelevät komponentit ovat adapteri (adapter), protokolla (protocol), multiplekseri (mux) ja tehdas (factory). Näillä on myös yhteinen nimittäjä, eli ne kaikki voivat toimia tietokanavana. Tietokanavalla on kyky kuljettaa tietoa toisesta päästä toiseen päähän (alarajapinnalta ylärajapinnalle ja toisinpäin). Lisäksi tietokanavia voidaan kytkeä kiinni toisiinsa, yleensä yksi kumpaankin päähän kaksisuuntaisesti.

Adapteria käytetään yleensä kytkemään ohjelmisto johonkin rajapintaan, vaikka sokettirajapintaan. Protokolla kuvaa jonkin protokollan sisältöä ja toimintaa. Protokollalla on tiloja (yksi tai useampi) ja muita parametreja. Multiplekseri voi kytkeä monta tietokanavaa B-puolelleen, yleensä protokollia, jotka voivat kuvata vaikka eri yhteyksiä tai linkkejä käsiteltäessä ATM-merkinantoa. Tehdasta käytetään luomaan ajon aikana uusia instansseja, yleensä protokollista, jotka ATM-merkinannossa ovat yleisimmillään uusia yhteyksiä (esim. B-ISUP AE). Tehtaalla on tallessa parametrina prototyyppi kopioitavasta instanssista.

Tärkeimmät tietoa esittäviä komponentteja ovat viestit ja kuljettajat. Viesti (messenger) on yleensä tietoa sisältävä olio, esimerkiksi protokollan pdu, ja

kuljettaja (transporter) tämän tiedon kuljettava olio conduitien välillä. Ajastimet (timers) ovat myös viestejä joita protokollien tilakoneet voivat ottaa vastaan aiheuttaen jonkin toiminnan.



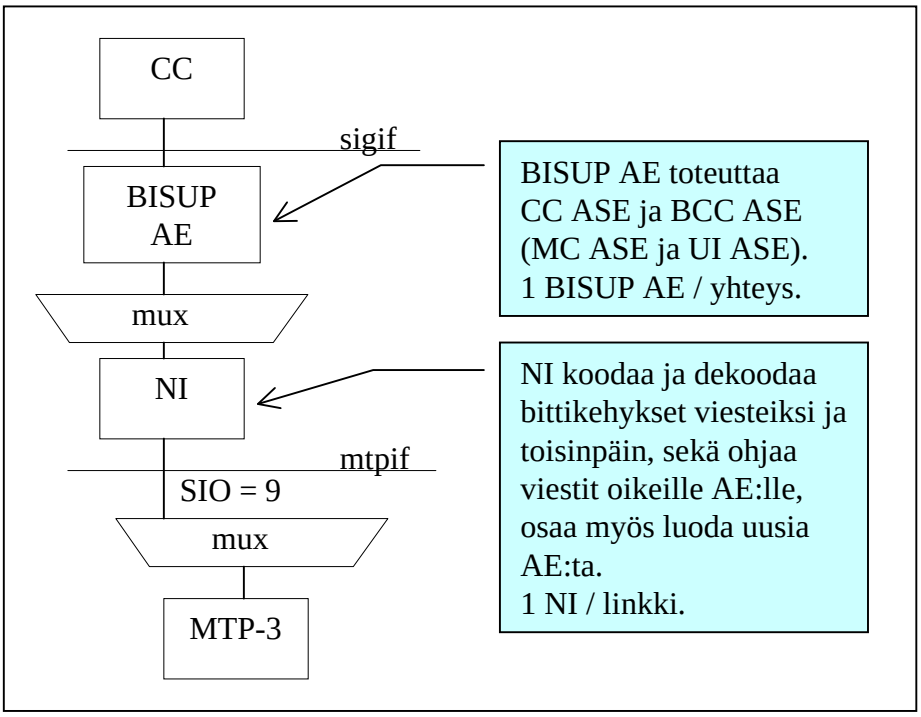
Kuva 5. OVOPS++:n pääkomponentit.

4 TOTEUTUS

4.1 Lähtökohta

Tässä vaiheessa kaikki tarvittava pohjatieto protokollasta ja sen ympäristöstä on hankittu. Standardit on käyty läpi, ja kaikki tärkeä tieto merkattu ylös. Toteutusympäristöön, eli työkaluihin, ja sen antamiin mahdollisuuksiin on myös tutustuttu. Tässä vaiheessa tiedetään tarkalleen mitä protokollasta toteutetaan. B-ISUP:n yläpuolinen sovellus CC, sekä alapuolinen siirtopalvelua tarjoava protokolla MTP-3 on toteutettu, sekä rajapinnat niiden välillä. Puuttuu vielä lopullinen protokollan toteutustapa käytössä olevalla protokollasovelluskehyksellä.

B-ISUP:n sovelluskokonaisuus siis sisältää monia eri toimintoja, jotka on hajautettu eri toiminnallisiin lohkoihin (kuva 2.). Sovelluspalveluelementit (BCC ASE ja CC ASE), niiden kontrollifunktio (SACF) sekä NI muodostavat työssä toteutettavan kokonaisuuden. NI on vastuussa joistain viestien virheiden tarkistuksista (mm. pituus), sekä viestien ohjaamisesta oikeille B-ISUP AE:lle ja tarvittaessa uuden AE:n luomisesta. NI:n toiminnot onkin selkeää ja perusteltua toteuttaa omaan protokollaan yhdellä tai useammalla tilakoneella. Lisäksi osa SACF:n toiminnoista voisi olla hyödyllistä sijoittaa samaan NI-protokollaan, kuten bittikehysten koodaus ja tulkitseminen. Jäljellä on B-ISUP AE. Pitää siis määrittää toteutustapa AE:lle, joka kattaa SACF:n niiltä osin mitä ei NI:ssä toteuteta sekä eri sovelluspalveluelementit ja näiden välisen kommunikoinnin (mm. tarvittavat sisäiset signaalit). SACF:n osuudesta jää toteutettavaksi vielä primitiivien koordinointi, kasaaminen ja hajauttaminen, sekä sovelluspalveluelementtien hallinta (mm. luominen ja tuhoaminen).



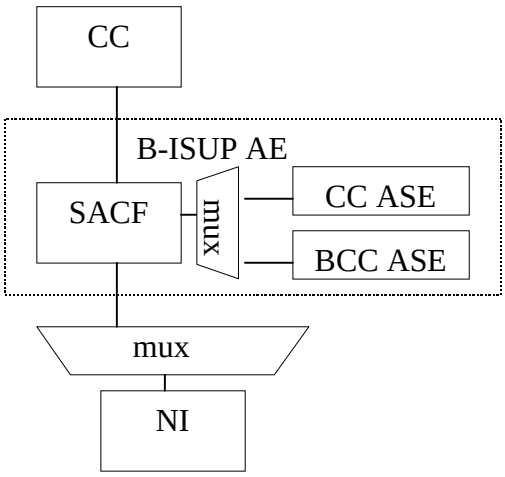
Kuva 6. Toteutuksen osakokonaisuudet.

OVOPS++ antaa monta mahdollisuutta AE:n toteutukseen. Voidaan muodostaa jokaisesta toiminnallisesta lohkosta erillinen protokolla ja kytkeä nämä monella tapaa toisiinsa, tai voidaan myös sulauttaa kaikkien lohkojen toiminnat yhteen laajempaan protokollaan. Löydettiin kolme toteutuskelpoista vaihtoehtoa, jotka esitellään seuraavaksi.

4.2 Toteutusvaihtoehto 1

Standardi Q.2764 määrittelee lohkokaaviona sovelluspalveluelementit erillisinä sekä SACF:n hallitsemaan niitä ja koordinoimaan liikennettä sovelluksessa. Looginen tapa on siten miettiä ratkaisua, jossa jokainen lohko on oma protokollansa omilla tilakoneillaan. Etenkin kun standardi määrittää jopa tilakoneet näille erillisille lohkoille. Eli siinä tapauksessa toteutus olisi helppo ja selkeä sovelluspalveluelementtien osalta. Niiden tilakoneet ovat SDL-kaaviona (Specification and Description Language), sekä toiminnat pääpiirteittäin selostettu tekstissä. Tämä toteutustapa mahdollistaisi myös

helpon tavan lisätä mahdolliset jatkototeutettavat sovelluspalveluelementit (MC ASE ja UI ASE). SACF olisi tässä toteutuksessa suuressa roolissa, ja sen toiminta on määritelty standardissa epämääräisesti ja vähemmän standardilla tavalla. Suurin vaikeus tässä olisi SACF:n toiminta viestien kasaamisessa, eli sen täytyisi tietää milloin yksi sisäinen signaali yhdeltä ASE:lta tai kaksi tai useampia signaaleja usealta ASE:lta aiheuttaisi yhden primitiivin lähetyksen CC:lle tai NI:lle. Sekä toisinpäin alhaalta tai ylhäältä tulevien primitiivien hajauttaminen yhdelle tai useammalle ASE:lle yhdellä tai useammalla signaalilla. Bittikehysten koodaus voisi tässä toteutuksessa olla joko SACF:ssä tai NI:ssä.

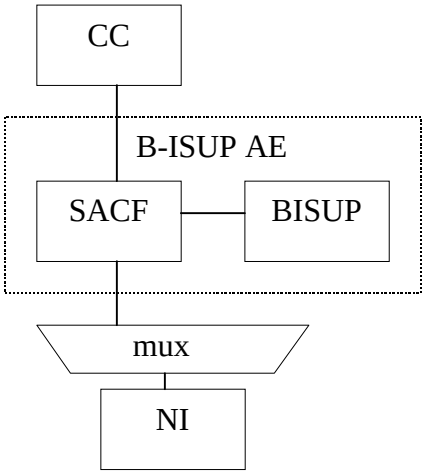


Kuva 7. Toteutusvaihto 1.

4.3 Toteutusvaihtoehto 2

Koska sovelluspalveluelementtien toiminnallisuus on varsin pientä, sekä synkronointi suurempi ongelma SACF:n toiminnallisuudessa, mietittiin toteutusvaihtoehtoa, jossa kaikki sovelluspalveluelementit toteutetaan yhdeksi protokollaksi. Tämän protokollan nimi olisi BISUP-protokolla (class bisupProtocol). Näin vältetään synkronointiongelmat, sekä saadaan karsittua sisäisten signaalien määrää huomattavasti. Nyt suurimmaksi työksi

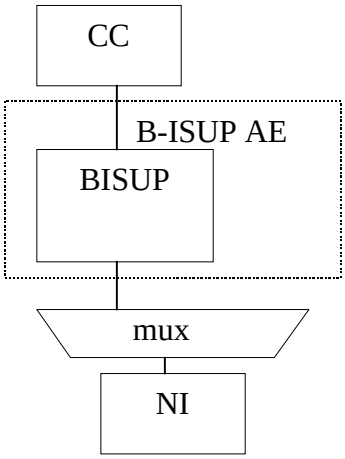
tulee eri sovelluspalveluelementtien tilakoneiden ja toimintojen yhdistäminen yhteen protokollaan ja sen tilakoneisiin. Myös tässä toteutuksessa bittikehysten koodaus voi olla joko SACF:ssä tai NI:ssä.



Kuva 8. Toteutusvaihtoehto 2.

4.4 Toteutusvaihtoehto 3

Edellinen vaihtoehto on tavallaan välimuoto tämän ja ensimmäisen välillä. Jos eri sovelluspalveluelementit eivät yksittäisenä toteuta paljoa toimintaa ja SACF:n toteutus on hankalaa, mutta sen toiminta vähenee kun ei tarvitse enää koordinoida kuin yhdelle protokollalle, miksi ei toteuttaa niitä kerralla yhdellä protokollalla. Edellisessä toteutuksessa jäi SACF:lle vain hyvin yksinkertainen toiminta, eli suoraviivainen primitiivien koordinointi yläpuolisen CC:n, protokollan ja alapuolisen NI:n välillä, jos bittikehysten koodaus suoritetaan NI:ssä. Ottamalla SACF pois erillisenä toimintona, ei tarvita juuri lainkaan lisää toimintoa BISUP-protokollaluokkaan ja sen tilakoneisiin, mutta saadaan paljon koodia pois.



Kuva 9. Toteutusvaihtoehto 3.

4.5 Valinta ja perustelut

Ensimmäinen vaihtoehto kuvaisi parhaiten standardin esittämän toiminnallisuuden jaon. Mutta siinä olisi koodattava todella monimutkainen SACF primitiivien ja signaalien synkronointiin ja käsittelyyn, sekä vaadittaisiin todella suuri määrä AE:n sisäisiä signaaleja. Nämä sisäiset rajapinnat määriteltiin standardissa vain käsitteellisiksi, ei testattaviksi. Lisäksi merkittävä tekijä oli yksittäisten toimintojen vähyys yksittäisessä sovelluspalveluelementissä.

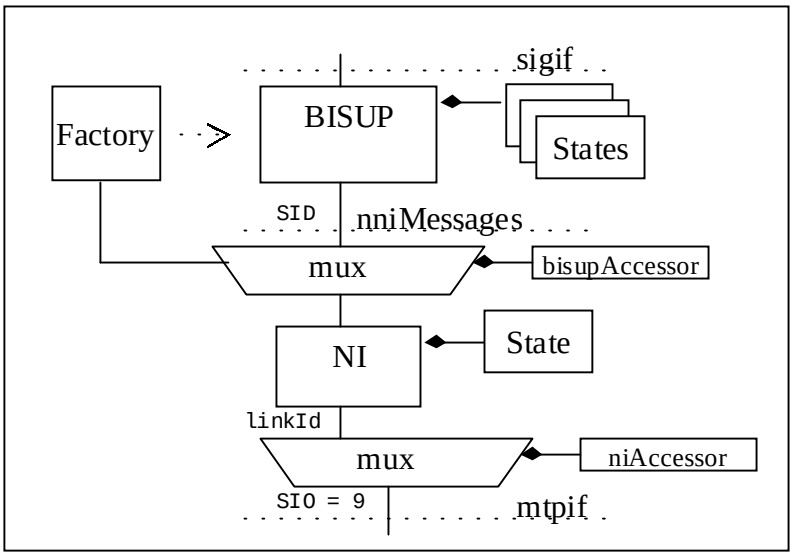
Toisessa vaihtoehdossa otettiin askel kohti yksinkertaisempaa lohkojakoa. Muodostettiin suurempi kokonaisuus pienistä osista. Suurin ongelma olisi ollut tilakoneiden hahmotus. Pitäisi muodostaa uudet yhden protokollan tilakoneet, joita ei standardissa kuvata, vastaamaan standardissa kuvattuja useita erillisiä tilakoneita. Tämä ei kuitenkaan olisi kovinkaan suuri työ, koska esim. BCC ASE:n tilakoneet toimivat vain yhteyden muodostuksen alkuvaiheessa ja purkamisessa, jossa alkuvaihe tarkoittaa lähinnä vahvistuksen saamista IAM-viestin (Initial Address Message) lähetykseen. CC ASE:n tilakoneet toimivat taas vain yhteyden muodostuksessa.

Valituksi tuli kolmas, viimeisenä esitelty, vaihtoehto. Toinen vaihtoehto tuntui ajavan ensimmäisen ohi, ja kolmas vaihtoehto vain yksinkertaistaisi koko toteutusta lisää, mm. vähentäisi tarvittavien signaaliluokkien määrää ja koodia muutenkin. Suurin työ tässäkin olisi muihin ratkaisuihin verrattuna yhdistettyjen tilakoneiden määrittäminen CC ASE:n ja BCC ASE:n tilakoneiden pohjalta. Mutta se oletettavasti olisi pienempi työ kuin SACF:n monimutkaisen koordinoinnin toteutus.

5 TOTEUTUSYKSITYISKOHTIA

5.1 Johdanto

Seuraavaksi käydään läpi lopullinen toteutustapa, sekä joitain toteutuksen yksityiskohtia. Samalla käydään läpi kaikki MTP:n tarjoamat primitiivit, tuetut B-ISUP:n viestit eli PDU:t (Protocol Data Unit) ja viestirajapinta CC:lle. Mahdolliset jatkokehitettävät ominaisuudet, kuten loput PDU:t, jätetään tässä huomioitta.



Kuva 10. TOVE-projektin B-ISUP:n toteutus.

5.2 B-ISUP AE

Toteutuksessa BISUP-protokolla (class bisupProtocol) on peritty pf:n pfProtocol-luokasta. Se toteuttaa osan SACF:n toiminnasta, sekä kokonaisuudessaan CC ASE:n ja BCC ASE:n toiminnat. Tarvitaan yksi BISUP-protokolla jokaista yhteyttä kohden ja aina uuden yhteydenpyynnön tullessa tehdas (factory) luo uuden instanssin. Jokainen BISUP-instanssi on identifioitu SID-arvolla (Signalling Identifier), jonka multiplekseri NI:n yläpuolella asettaa asentaessaan tehtaan antaman uuden instanssin itsensä yläpuolle. Saadessaan ensimmäisen viestin (IAM), BISUP-protokolla saa

vastapuolen SID:n, jonka avulla se identifioi jatkossa kohteen. Ensimmäisessä vastausviestissä (IAA, IAM Acknowledge Message) vastaavasti BISUP lähettää oman SID:nsä vastapuolelle. Näin molemmilla on oma tunnus (engl Originating Signalling Identifier, OSID), sekä vastapuolen tunnus tiedossa (engl. Destination Signalling Identifier, DSID).

Tilakoneiden yhdistäminen onnistui ongelmitta. Tuloksena oli yksitoista tilaa bisupProtocol-luokalle (liite 1), joilla saatiin kuvattua standardissa määritetyt kaksitoista BCC ASE:n tilaa ja kahdeksan CC ASE:n tilaa. BCC ASE:lle määritetty suuri tilakoneiden määrä johtuu standardin tavasta merkitä eri tilat sekä sisääntulevalle että uloslähtevälle linkille, vaikka ne käytännössä ovatkin aivan samat.

5.3 NI

NI:n toiminnan muodostaa niProtocol-luokka, joka on myös peritty pfProtocol-luokasta, kuten kaikki TOVE:n protokollat. NI:llä on vain yksi tila. NI koodaa bittikehykset ja ohjaa niistä saadut PDU:t oikealle bisupProtocol-instanssille (DSID). Multiplekseri NI:n yläpuolella käyttää bisupAccessor-luokkaa saadakseen viestistä ulos DSID-arvon pystyen siten ohjaamaan viestin oikeaan paikkaan. Tarvittaessa NI pystyy luomaan uuden bisupProtocol-instanssin tehtaan avulla. Ylhäältä tulevat PDU:t NI kokoaa bittikehyksiksi ja lähettää ne MTP:lle. Jokaista linkkiä kohden on yksi niProtocol-instanssi. MTP:n yläpuolella oleva on kaksi päällekkäistä multiplekseriä. Alempi ohjaa liikennettä MTP:n käyttäjien välillä (SCCP, ISUP, B-ISUP yms.) ja ylempi määrittää eri B-ISUP-linkit kytkimessä.

5.4 Primitiivit

Seuraavassa taulukossa esitetään tuetut primitiivit. Taulukko on osa TOVE-projektissa syntynyttä BISUP-dokumenttia. Dokumentointikielenä projektissa toimi englanti. Primitiivien suunta on esitelty taulukossa seuraavasti:

1) Primitiivit B-ISUP:lta CC:lle.

2) Primitiivit CC:ltä B-ISUP:lle.

3,4) PDU:t bisupProtocol:n ja niProtocol:n välissä, kuvaavat myös tavallaan peer-rajapinnan.

5) MTP-3:lta tulevat primitiivit.

6) MTP-3:lle menevät primitiivit.

No	Primitive/Message	Function
1.	sigSETUPind	Indicates receive of IAM pdu
	sigSETUPconf	Indicates receive of ANM pdu
	sigPROCEEDINGind	Indicates receive of IAA pdu
	sigALERTINGind	Indicates that all address information have been
	sigRELEASEind	Indicates receive of REL pdu
	sigRELEASEconf	Indicates that signalling is terminated (RLC or IAR)
2.	sigSETUPreq	Causes call setup to begin → IAM will be sent
	sigSETUPresp	Sent after call is ready → ANM will be sent
	sigPROCEEDINGreq	After call is proceeding → IAA will be sent
	sigALERTINGreq	All address information reveived → ACM will be sent
	sigRELEASEreq	Causes a call release process to be started → REL
	sigRELEASEresp	Sent after releasing call is OK → RLC (or IAR)
3, 4	nniANMpdu	Answer message
	nniACMpdu	Address Complete Message
	nniCPGpdu	Progress Message
	nniIAApdu	IAM Acknowledge Message
	nniIARpdu	IAM Reject Message
	nniIAMpdu	Initial Address Message
	nniRELpdu	Release message
	nniRLCpdu	Release Complete message
5.	mtpTRANSFERind	Data transfer indication from MTP3
	mtpSTATUSind	Indicates unavailibility of the signalling route.
	mtpPAUSEind	Unable to transfer message to the destination.
	mtpRESUMEind	Able to resume message transfer.
6.	mtpTRANSFERreq	Data transfer request primitive to MTP3

Taulukko 1. Tuetut primitiivit ja PDU:t. /11/

6 TESTAUS JA JATKOKEHITYS

Kolme erilaista toteutusvaihtoehtoa OVOPS++-ympäristöön esiteltiin ja yksi päätyi valituksi perusteluineen. Dokumentointi alkoi heti kun toteutuksen suunnitelma valmistui, eli ennen koodausta. Kun toteutettava protokolla on dokumentoitu myös toteutuksen osalta ennen koodausta, turvataan viimeistelty suunnittelu. Dokumentointia jatketaan tietysti siitä lähtien koko lopputoteutuksen ajan päivittäen mahdolliset muutokset. Toteutuksen jälkeen pyrittiin testaamaan protokolla joillain omilla testiohjelmilla. Tällä testailulla voidaan vain varmistaa yleinen toimivuus, sekä karsimaan joitain selkeitä virheitä. Lähinnä merkittävintä testaamista oli bittikehysten koodauksen ja kokoamisen testaaminen erilaisilla itse rakennetuilla summittaisilla testikehyksillä. Protokollan jatkokehitys riippuu havaittavasta tarpeesta projektissa. Seuraava vaihe on saada yhteentoiminta Q.2931-protokollan kanssa samassa ohjelmistossa. Se paljastanee virheitä johtuen puutteellisesta testauksesta. Lisää viestejä ei ole kuitenkaan tarkoitus enää tukea, vaan toteutettujen viestien pitäisi riittää hyvin tukemaan esim. Q.2931-käyttäjämerkinantoa verkon yli.

7 JOHTOPÄÄTÖKSET

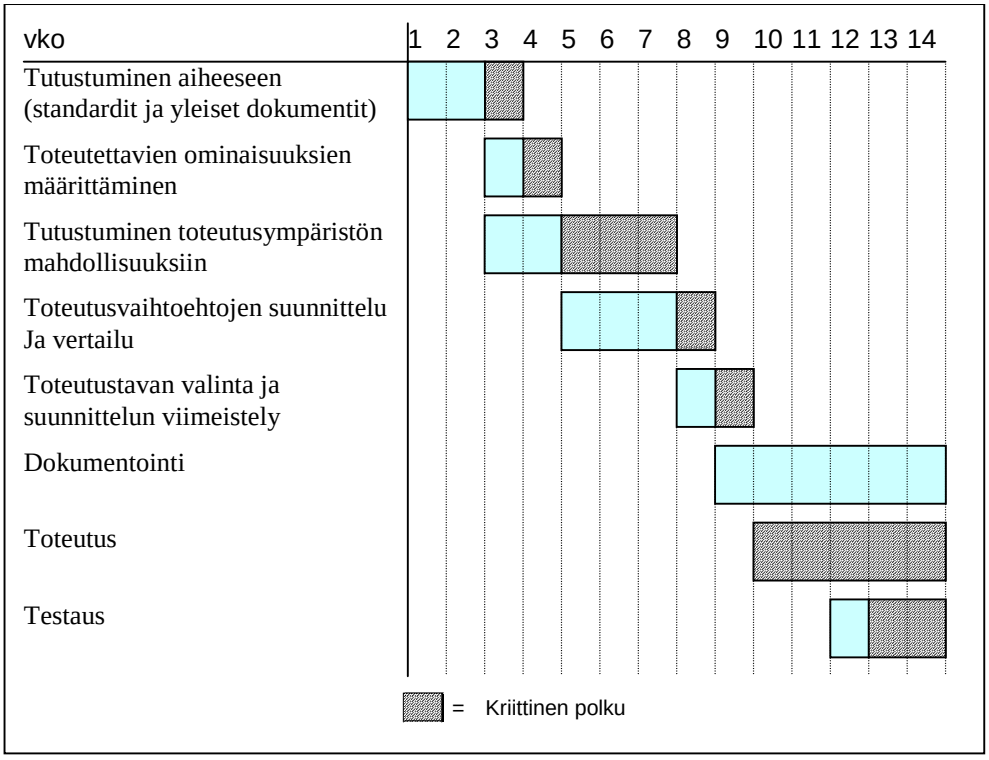
Tietoliikenneprotokollan toteuttaminen on vaativa tehtävä ja sisältää paljon enemmän kuin pelkistetysti ilmaisten standardin siirtämisen koodin muotoon. Standardi määrittelee protokollan yleensä tarkasti, jotta se toimisi johdonmukaisesti tilanteessa kuin tilanteessa, aiheuttamatta suuria yllätyksiä. Standardi ei kuitenkaan puutu toteutuksen kaikkiin yksityiskohtiin, koodaustapaan, ja jättää paljon toteuttajan mietittäväksi. Standardi kertoo tietyt asiat, kuinka tiettyyn viestiin pitää vastata tai mitä tehdään jos jokin viesti ei sisällä tiettyä informaatiota. Standardi erottaa yleensä tietyn kerroksen protokollan moniin toiminnallisiin lohkoihin. Kuinka nämä lohkot näkyvät koodissa, etenkin oliopohjaisessa, riippuu koodin tekijästä. Oliopohjainen suunnittelu antaa suorastaan rajattomat mahdollisuudet suunnitella funktionaalisten toimintojen erottelu.

Protokollan kehityksessä ja suunnittelussa on monta eri vaihetta. Ensimmäinen vaihe on teoriavaihe, aiheeseen tutustuminen. Se ei sisällä pelkästään toteutettavan protokollan tutkimista, vaan myös sen ympäristön ja kokonaisuuden hahmottamista. Samalla kun pääsee paremmin aiheeseen sisälle, suunnittelija pystyy hahmottamaan protokollan piirteitä tärkeysjärjestykseen ja poimimaan sieltä pakolliset ominaisuudet, jollei niitä tarkasti missään ole aiemmin määritelty.

Protokollan ja sen toteutettavien ominaisuuksien selvennyttyä, alkoi tutustuminen käytössä oleviin työkaluihin. Tässä tapauksessa työkalut käsittivät protokollan kehitysympäristön OVOPS++, C++-ohjelmointikielen antamat mahdollisuudet, sekä erilaisten suunnittelumallien (design patterns) mahdollinen lisähyödyntäminen. Suunnittelumallithan ovat jo osa OVOPS++-sovelluskehystä, mutta niitä ei pidä siihen unohtaa, vaan koko ajan on tarkoituksena löytää uusia tapoja hyödyntää niitä. Tässä tapauksessa tekijällä oli jo hieman aiempaa kokemusta ympäristöstä, mikä lyhensi

tutustumiseen vaadittavaa aikaa. Enemmän aikaa vaativa tehtävä olikin tässä työssä pyrkiä löytämään paras mahdollinen tapa ratkaista ohjelmallisesti standardien määrittelemien toiminnallisten lohkojen toteutus.

Projektisuunitelmilla on ikävä taipumus olla liian tiukkoja. Niin kävi tässäkin. Eniten aikaa vei, ja enemmän kuin oli tarkoitus, aiheeseen tutustuminen. Toisaalta taas huolellisella aiheeseen tutustumisella ja suunnittelulla voidaan säästää paljon aikaa toteutuksessa. Lisäksi haittana oli toisten päällekkäisten töiden tekeminen. Aina pitäisi keskittyä vain yhteen asiaan kerrallaan. Yleisesti ottaen aikataulussa kuitenkin pysyttiin. Protokollan toteutus ei silti ole välttämättä ohi, vaan kehitys mahdollisesti jatkuu ja samoin testaus.



Kuva 11. Alustava aikataulu.

LÄHTEET

1. Koponen, Petteri. Puro, Vesa-Matti. Räsänen, Juhana. Nummisalo, Pasi. Martikainen, Olli. Distribution of Switch Control. Proceedings of the Workshop on Broadband Network Intelligence. Espoo, Finland. 1996.
2. Raatikainen, Pertti. Analysis and Implementation of a High-Speed Packet Switching Architecture - the Frame Synchronized Ring. Doctorate thesis, Helsinki University of Technology, Espoo (June 1996).
3. Händel, Rainer. Huber, Manfred. Schröder, Stefan. ATM Networks. Addison-Wesley. 1998. ISBN 0-201-17817-6.
4. International Telecommunications Union, ITU-T. URL: http://www.itu.int/.
5. ITU-T. Recommendation Q.2761. Broadband Integrated Services Digital Network (B-ISDN) - Functional description of the B-ISDN User Part (B-ISUP) of Signalling System No.7. 1995.
6. ITU-T. Recommendation Q.2761. Broadband Integrated Services Digital Network (B-ISDN) - General functions of messages and signals of the B-ISDN User Part (B-ISUP) of Signalling System No.7. 1995.
7. ITU-T. Recommendation Q.2761. Broadband Integrated Services Digital Network (B-ISDN) - Signalling Systems No.7 B-ISDN User Part (B-ISUP) - Formats and codes. 1995.
8. ITU-T. Recommendation Q.2761. Broadband Integrated Services Digital Network (B-ISDN) - Signalling Systems No.7 B-ISDN User Part (B-ISUP) - Basic call procedures. 1995.
9. Huni, H., Johnson, R., Engel, R.. A Framework for Network Protocol Software. OOPSLA 1995.
10. Gamma, E., Helm, R., Johnson, R., Vlissides, J. Design Patterns: Elements of Reusable Object-Oriented Software. Addison-Wesley Publishing Company. 1995. ISBN 0-201-63361-2.
11. Raatikainen, Sami. TOVE BISUP. TKK. 1999.

LIITE 1. Toteutetut tilat. /11/

- Idle (both sides)
  No connection present, or reserved resources.

- Await IAA (outgoing side)
  IAM sent to the other side. Waiting for resource acceptance (IAA) or rejection (IAR).

- Await ACM (outgoing side)
  All address information has been sent to the other side. Waiting for indication that the connection has been accepted (ACM/ANM).

- Await ANM (outgoing side)
  The other side has informed that the connection is accepted. Waiting for an answer (ANM) to establish the connection.

- Await Accepted req (incoming side)
  IAM received, setup indication sent to application (Call Control). Waiting for resource acceptance or rejection.

- Await Address Complete req (incoming side)
  Waiting for the application to confirm that all the address information has been received, and the connection has been accepted, or possibly an immediate answer.

- Await Answer req (incoming side)
  The application informed that all the address information needed has been received, and the other has been informed with ACM. Waiting for a final answer from the application.

- Call Answered (both sides)
  Call has been answered, and the connection established.

- Await RLC (both sides)
  This side sent REL pdu to the other side. Waiting for reply.

- Await Release resp (both sides)
  REL pdu sent to the other side, waiting for RLC to release all resources and to terminate the instances.

- REL Collision (both sides)
  Both sides sent REL pdus at the same time. Confirmation needed from the other side (RLC), and the application (release resp).