TAMPEREEN TEKNILLINEN KORKEAKOULU
Tietotekniikan osasto

OLLI-PEKKA SAVIA

DVB-vastaanottimen laitteistorajapinnan Java-toteutus

DIPLOMITYÖ

Aihe hyväksytty osastoneuvoston kokouksessa 8.9.1999
Tarkastajat: Prof. Ilkka Haikala
Prof. Seppo Kalli

Alkulause

Tämä diplomityö on syntynyt työskennellessäni Tampereen teknillisen korkeakoulun Digitaalisen median instituutissa. Työn valmistumista ovat edesauttaneet Future TV -projektissa työskennelleet tutkijakollegat, joiden asiantunteva apu oli usein tarpeen. Tutkijakollegat ansaitsevat myös kiitoksen kannustavasta työilmapiiristä, joka auttoi työn valmistumista ajallaan. Haluan lisäksi kiittää työni ohjaajia Prof. Ilkka Haikalaa ja Prof. Seppo Kallia saamastani ohjauksesta ja monista työtä koskevista neuvoista. Kiitokset myös DI Jukka Rakkolalle, jonka kommentit koskien työn kirjallista osaa, auttoivat poistamaan monia asiavirheitä. Erityiskiitokset ansaitsee YTM Nina Urbano, joka tarkkaavaisesti tarkasti ja korjasi työn kieliasun.

Tampereella 24. tammikuuta 2000

Olli-Pekka Savia
Kraatarinkatu 9–11 B 12
33270 Tampere

Sisältö

Tiivistelmä

TAMPEREEN TEKNILLINEN KORKEAKOULU
Tietotekniikan osasto
Ohjelmistotekniikka
SAVIA, OLLI-PEKKA: DVB-vastaanottimen laitteistorajapinnan Java-toteutus
Diplomityö, 48 sivua, 2 liitesivua
Tarkastajat: Prof. Ilkka Haikala ja Prof. Seppo Kalli
Rahoittaja: TEKES
Tammikuu 2000
Avainsanat: Digitaalinen TV, MPEG-2, DVB, Java, Multimedia Home Platform

Suomi on ensimmäisten maiden joukossa, joka ottaa käyttöön maanpäällisen digitaalisen TV-lähetysverkon. Digitaalisen siirtotekniikan käyttö mahdollistaa monien vuorovaikutteisten palvelujen toteuttamisen TV-ympäristössä. Myös osa Internetin vuorovaikutteisista palveluista on mahdollista sovittaa käytettäväksi TV:n kautta.

Tässä työssä toteutettu digitaalisen TV-vastaanotinkortin Java-rajapinta on osa kansallisen Future TV -projektin tutkimusta. Future TV -projektin tehtävänä on tutkia ja kehittää digitaalisen television vuorovaikutteisia palveluja sekä niihin liittyviä tekniikoita. Yksi tutkimuksen päätavoitteista on tutkia vuorovaikutteisten palvelujen ja niihin liittyvän datan välitystä yleisjakeluverkossa.

Tällä hetkellä ei ole saatavilla sopivia digitalisia vastaanotinlaitteistoja, joissa vuorovaikutteisten palvelujen tutkiminen ja käytännön testaaminen onnistuisi. Tästä syystä Future TV -projektissa on kehitetty PC-pohjainen testialusta, joka pohjautuu tässä työssä kehitettyyn Java-rajapintaan. Java-rajapinta mahdollistaa digitaalisen MPEG-2-lähetteen vastaanoton PC:llä Java-ohjelmasta käsin. Testialustan avulla on tutkittu mm. data- ja objektikarusellien toimintaa ja vuorovaikutteisten Java-sovellusten välittämistä digitaalisen TV-lähetteen mukana.

Abstract

TAMPERE UNIVERSITY OF TECHNOLOGY
Department of Information Technology
Software Systems Laboratory
SAVIA, OLLI-PEKKA: A Java Implementation of a DVB Receiver Hardware Interface
Master of Science Thesis, 48 pages, 2 enclosure pages
Examiners: Dr. Ilkka Haikala and Dr. Seppo Kalli
Funding: TEKES
January 2000
Keywords: Digital TV, MPEG-2, DVB, Java, Multimedia Home Platform

Finland is among the first countries to start using digital terrestrial broadcasting networks. Digital broadcasting enables many interactive services in the TV environment. It also enables the Internet's interactive services to be accessed via TV.

This thesis represents the implementation of a Java inteface for a digital TV receiver card. The thesis was carried out as a part of a national Future TV project. The aim of the Future TV project is to study and develop technical concepts involved in the interactive services of digital television. One of the main subjects is to study the delivery of interactive services and related data in the broadcasting networks.

Currently, there are no suitable digital receivers available for research and practical testing purposes. Therefore, the Future TV project has developed a PC based testing platform which is based on the Java interface developed in this thesis. The Java interface allows the reception of digital MPEG-2 transmissions from a Java application. The platform has been used to study functions of data and object carousels and the delivery of interactive Java applications in the digital broadcasting networks.

Lyhenteet

AIT	Application Information Table
API	Application Programming Interface
ATM	Asyncronous Transfer Mode
AWT	Abstract Window Toolkit
BAT	Bouquet Association Table
BER	Bit Error Rate
CA	Conditional Access
DSM-CC	Digital Storage Media–Command and Control
DVB	Digital Video Broadcasting
DVB-C	Digital Video Broadcasting Cable
DVB-J	Digital Video Broadcasting Java
DVB-S	Digital Video Broadcasting Satellite
DVB-T	Digital Video Broadcasting Terrestrial
DVD	Digital Video Disk
EIT	Event Information Table
EPG	Electronic Program Guide
ES	Elementary Stream
HTML	Hyper Text Markup Language
IP	Internet Protocol
JDK	Java Development Kit
JNI	Java Native Interface
LLC	Logical Link Control
MAC	Medium Access Control
MHP	Multimedia Home Platform
MPE	Multi Protocol Encapsulation
MPEG	Moving Picture Experts Group
NIT	Network Information Table
NSAP	Network Service Access Point
PAL	Phase Alternating Line
PAT	Program Association Table
PCMCIA	PC Memory Card Interface Association
PES	Packetized Elementary Stream
PID	Packet Identification
PMT	Program Map Table
PS	Program Stream
PSI	Program Specific Information
RF	Radio Frequency
RGB	Red, Green, Blue
RTS	Running Status Table
SCART	Syndicat des Constructeurs d'Appareils Radio Recepteurs et Televiseurs
SDT	Service Description Table
SI	Service Information
ST	Stuffing Table
STC	System Time Clock
TDT	Time and Date Table
TS	Transport Stream
UML	Unified Modelling Language
VM	Virtual Machine
VST	Version Summary Table

Luku 1

Johdanto

Televisioverkkojen digitalisointi ja digitaalisen lähetystekniikan käyttöönottaminen mahdollistaa entistä tehokkaamman lähetyskaistan hyödyntämisen. Näin on mahdollista välittää entistä terävämpää kuvaa ja parempilaatuista ääntä. Näitä tärkeämpi digitaalitekniikan tuoma uusi ominaisuus on kuitenkin joustava lisäinformaation lähettäminen ohjelmalähetteen mukana. Televisio-ohjelmiin voidaan liittää monimuotoista lisämateriaalia kuten esimerkiksi monikielisiä tekstityksiä ja ääniraitoja, animaatioita ja muuta grafiikkaa sekä erilaisia vuorovaikutteisia palveluja. Digitaalinen lähetystekniikka mahdollistaa osittain myös Internet-palvelujen välittämisen yleisjakeluverkossa.

Digitaalisen televisiovastaanottimen rakenne eroaa huomattavasti perinteisestä analogiavastaanottimesta. Itse asiassa digitaalinen vastaanotin muistuttaa rakenteeltaan enemmän tietokonetta kuin perinteistä televisiota. Vaikka analogiavastaanottimissa on jo pitkään ollut mikroprosessori ohjaamaamassa laitteen toimintoja, digitaalivastaanottimessa prosessorilla on huomattavasti tärkeämpi rooli.

Digitaalisessa vastaanottimessa oleva prosessori, lisämateriaalin välitys ohjelmalähetteen mukana sekä vastaanottimeen liitetty paluukanava mahdollistavat aivan uudenlaisten lisäpalvelujen toteuttamisen. Katsojalle voidaan tarjota ohjelmalähetteen mukana — tai hän voi paluukanavaa pitkin ladata — sovelluksia, joita vastaanottimen prosessori suorittaa. Tällaisia sovelluksia voivat olla esimerkiksi sähköiset ohjelmaoppaat, pelit, sähköpostiohjelma ja Internet-selain. Tulevaisuudessa myös teksti-TV saa uuden ilmeen ja sen käyttö on entistä helpompaa.

Digitaalista televisiota ja siihen liittyviä palveluja on kehittelty eri yrityksissä jo muutamien vuosien ajan. Nyt ongelmaksi on muodostunut eri laitevalmistajien laitteistojen yhteensopimattomuus. Vastaanotinlaitteistot käyttävät eri prosessorityyppejä ja käyttöjärjestelmiä mistä seuraa, että yhdelle laitteistolle tehtyjä ohjelmistoja ei voi käyttää toisen laitevalmistajan laitteistossa. Tilanne on ongelmallinen sekä palvelujen tuottajille että kuluttajille. Palvelujen tuottajat joutuvat käyttämään ylimääräisiä resursseja sovittaakseen palvelunsa moneen eri laiteympäristöön ja vastaavasti kuluttajat joutuvat investoimaan useampiin vastaanotinlaitteistoihin, mikäli ohjelmistoja ei soviteta jokaiseen laiteympäristöön.

Estääkseen monien toistensa kanssa yhteensopimattomien suljettujen järjestelmien leviämisen, digitaalisia televisiojärjestelmiä Euroopassa standardoinut Digital Video Broadcasting (DVB) -organisaatio on ryhtynyt tekemään avoimia määrittelyjä digitaalisessa televisiossa käytettäville ohjelmistoille. Näiden määrittelyjen mukaiset ohjelmistot toimivat eri laitevalmistajien laitteistoissa.

Tässä työssä perehdytään DVB:n mukaisen digitaalisen televisiojärjestelmän toimintaan, digitaalijärjestelmässä käytettäviin päätelaitteisiin sekä määrittelyvaiheessa olevaan ohjelmistoalustaan: Multimedia Home Platformiin. Suomi on ensimmäisten maiden joukossa, joka on valinnut Multimedia Home Platformin digitaalisten vastaanotinlaitteistojensa ohjelmistoalustaksi.

Luvussa 2 tutustutaan Euroopassa käytetyn digitaalisen lähetystekniikan standardointiin ja toimintaan yleisellä tasolla. Luvussa käsitellään myös digitaalista vastaanotinta ja sen rakennetta. Luvun lopuksi tutustutaan digitaalisen lähetystekniikan mahdollistamiin uusiin palvelumuotoihin.

Luvussa 3 käsitellään protokollia, joita käytetään digitaalisen television äänen ja kuvan sekä uusien palvelujen toteuttamisessa tarvittavan lisädatan välittämiseen DVB:n mukaisessa televisiojärjestelmässä.

Luvussa 4 perehdytään Multimedia Home Platformin arkkitehtuuriin, ohjelmistorajapintoihin ja muihin ominaisuuksiin.

Luvussa 5 esitellään tässä työssä toteutettu digitaalisen satelliittivastaanotinkortin Java-rajapinta ja sen toteuttamiseen liittyviä yksityiskohtia.

Luku 2

Digitaalinen televisiojärjestelmä

2.1 Eurooppalainen standardi

Vuonna 1993 Eurooppaan perustettiin yhteistyöfoorumi kehittämään ja standardoimaan digitaaliset satellitti-, kaapeli- ja maanpäällisen TV-jakelun menetelmät. Tähän Digital Video Broadcasting nimen alla toimivaan yhteistyöfoorumiin kuuluvat alan tärkeimmät operaattorit, valmistajat ja tutkimuslaitokset. Tällä hetkellä DVB:hen kuuluu yhteensä yli 250 tahoa. [1]

DVB on onnistunut standardointityössään erittäin mallikkaasti. Standardiehdotukset digitaalisiksi jakelumenetelmiksi satelliitti- (DVB-S) ja kaapeli-TV-lähetykseen (DVB-C) valmistuivat vuosina 1994–1995 ja maanpäällistä jakelua (DVB-T) koskeva standardiehdotus valmistui keväällä 1996. [1]

Ensimmäisenä digitaaliset DVB-S-standardin mukaiset satelliittilähetykset aloitti Ranska syksyllä 1996. Maanpäällisten lähetysten edelläkävijänä toimi Englanti, jossa DVB-T-standardin mukaiset digitaalilähetykset alkoivat marraskuussa 1998.

Ruotsissa digitaalilähetykset alkoivat huhtikuussa 1999 ja Suomessa maanpäällisten digitaalilähetysten aloittaminen ajoittuu Sydneyn olympialaisten aikaan syksyllä 2000 [2]. Vuoden 2001 loppuun mennessä digitaalisen verkon näkyvyysalueen pitäisi kattaa 70% suomalaisista.

DVB:n standardoimissa järjestelmissä on paljon yhteisiä elementtejä, millä on pyritty tekemään mahdolliseksi lähetin- ja vastaanotinlaitteiden korkea integrointiaste ja siten alhaiset tuotantokustannukset. Eri siirtoteiden fysikaalisista ominaisuuksista johtuen standardit eroavat toisistaan lähinnä käytetyn modulaatiomenetelmän osalta. [3]

2.2 Digitaaliset televisiolähetykset

DVB:n määrittelemät digitaaliset lähetysjärjestelmät pohjautuvat MPEG-2-standardiin. MPEG-2-standardi sisältää menetelmät, joiden avulla audio- ja videosignaalit voidaan kompressoida tehokkaasti hyödyntämällä hallitusti silmän ja korvan vajavaisuuksia. Yhden analogisen lähetyksen viemällä kaistalla voidaan lähettää siirtotiestä ja halutusta kuvan- ja äänenlaadusta riippuen 4–15 MPEG-2 koodattua digitaalista lähetystä. [1]

MPEG-2-standardissa määritellään myös miten koodatut audio- ja videosignaalit sekä mahdollinen lisäinformaatio yhdistetään yhdeksi bittivirraksi eli multipleksoidaan. Multiplekserissä yhdistetty bittivirta myös paketoidaan ja jokaiseen pakettiin liitetään tunniste ja tarvittaessa aikaleima. Tunnisteiden perusteella vastaanotin osaa poimia lähetteestä haluamansa paketit ja aikaleimojen avulla audio- ja videopaketit voidaan tahdistaa keskenään oikein. Luvussa 3 käsitellään tarkemmin multipleksointia ja datan siirtoa DVB-järjestelmässä.

Edellä kuvattu prosessi on samanlainen riippumatta lähetykseen käytettävästä siirtotiestä. Multipleksoinnin jälkeen syntynyt pakettimuotoinen bittivirta moduloidaan ennen lähetystä kullekin siirtotielle sopivalla tavalla.

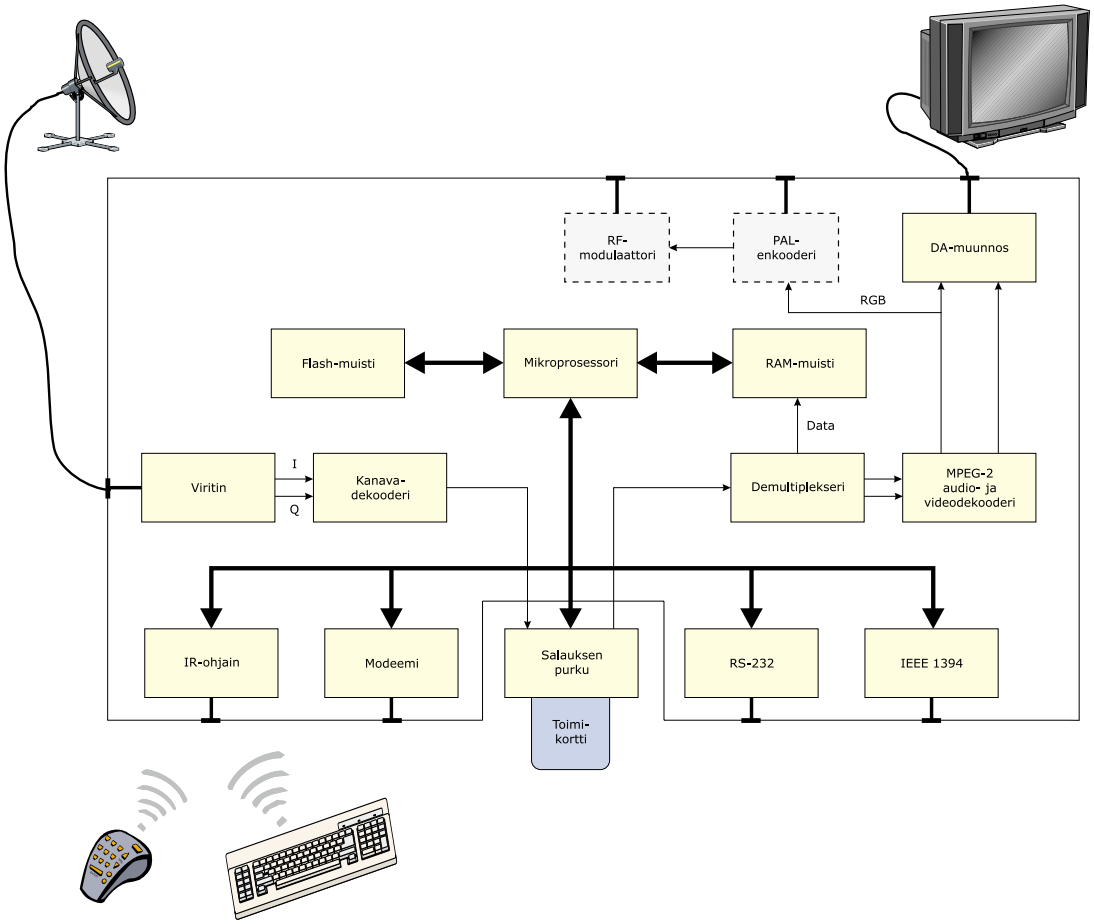
2.3 Digitaalisen televisiovastaanottimen rakenne

Digitaaliset vastaanottimet tulevat ainakin aluksi olemaan televisioon liitettäviä lisälaitteita eli ns. multimediapäätteitä (set-top box). Eri verkoissa toimivat vastaanottimet tarvitsevat hieman erilaiset vastaanottimet. Tulevaisuudessa voidaan myös valmistaa yhdistelmävastaanottimia, joissa on virittimet ja kanavadekooderit sekä satelliitti- että maanpäälisiin verkkoihin tai jopa kaikille kolmelle siirtotielle. [3]

Kuvassa 2.1 on esitetty digitaalisen vastaanottimen tärkeimmät lohkot: viritin, kanavadekooderi, salauksenpurkumoduuli, demultiplekseri, audio- ja videodekooderit, mikroprosessori muisteineen sekä tarvittavat liitännät. [3], [4]

Digitaalisessa vastaanottimessa virittimellä on sama tehtävä kuin analogisessa vastaanottimessa. Viritin valitsee lähetteestä halutun kanavan ja siirtää sen kantataajuudelle tai pienelle välitajuudelle. Seuraavaksi signaali demoduloidaan, jonka jälkeen virittimen lähdöstä saadaan analogiset I- ja Q-signaalit. Analogiset signaalit syötetään kanavadekooderille, joka muodostaa digitaalisen signaalin ja tekee tarvittavat virheenkorjausoperaatiot. [1]

Seuraavaksi virhekorjattu bittivirta syötetään salauksenpurkumoduulille. Salauksen purkamisessa tarvittava descrambler-piiri voi olla rakennettu joko irrotettavaan PCMCIA-moduuliin tai kiinteästi vastaanottimeen. Salauksen purkamisen



Kuva 2.1. Digitaalisen vastaanottimen lohkokaavio.

jälkeen bittivirta kulkee demultiplekserille, joka erottelee siitä audio-, video- ja datapaketit PID-tunnisteiden perusteella. [1]

Demultiplekseriltä saatavat audio- ja videopaketit syötetään MPEG-2-dekooderille, joka muodostaa paketeissa olevasta informaatiosta audio- ja videosignaalit. PAL-enkooderi ja RF-modulaattori muuttavat RGB-signaalin videosignaaliksi ja edelleen TV:n antenniliittimeen sopivaksi radiosignaaliksi. PAL-koodauksessa signaaliin syntyy runsaasti virheitä, jotka on digitaalitekniikalla pystytty poistamaan, joten PAL-enkooderin sisällyttäminen digitaaliseen vastaanottimeen on kyseena-

laista. Jotta set-top box voidaan liittää vastaanottimeen, jossa ei ole SCART-liitintä, PAL-enkooderin ja RF-modulaattorin sisällyttäminen on kuitenkin tarpeellista. Ajan myötä, kun vastaanotinlaitteisto uusiutuu, ne tulevat kuitenkin häviämään digitaalisista vastaanottimista. [1]

Demultiplekseri ohjaa vastaanotetut datapaketit mikroprosessorin käsiteltäväksi. Digitaalisessa vastaanottimessa mikroprosessoria käytetään laitteen toimintojen ohjaamisen lisäksi erilaisten sovellusohjelmien suorittamiseen. Nämä sovellusohjelmat voivat olla tallennettuna pysyvästi laitteen muistiin tai ne voidaan ladata ohjelmalähetteen mukana tai paluukanavaa pitkin. [1], [3]

Digitaaliset vastaanottimet rakennetaan yleensä “future proof” -periaatteella, jolla tarkoitetaan sitä, että kotipäätteen käyttöjärjestelmä ja järjestelmäohjelmisto voidaan jälkeenpäin päivittää lähettämällä uusi ohjelmaversio televisiolähetteen mukana [5]. Yhteispohjoismaisen NorDig-organisaation asettamissa vaatimuksissa pohjoismaissa käytettäville vastaanottimille ohjelmiston päivitys on pakollinen ominaisuus [6].

2.4 Digitaalisen television tarjoamat uudet palvelut

Pelkästään paremmalla kuvan- ja äänenlaadulla sekä uusilla kanavilla on vaikea saada kuluttajia innostumaan digitaalisesta televisiosta ja investoimaan uuteen vastaanotinlaitteistoon tuhansia markkoja. Digitaalisen television puuhamiehet ovatkin jo pitkään herätelleet kuluttajien kiinnostusta sen tarjoamilla uusilla palveluilla. Nämä uudet palvelut hyödyntävät datan välitystä televisiolähetteen yhteydessä sekä mahdollisuutta kytkeä vastaanotin paluukanavan avulla palvelun tarjoajaan.

Paluukanava mahdollistaa todellisten vuorovaikutteisten palvelujen toteuttamisen. Paluukanavayhteytenä voi toimia esimerkiksi tavallinen puhelinverkko, mobiiliverkko tai kaksisuuntainen kaapeli-TV-yhteys. Paluukanavan avulla televisiovastaanotinta voidaan käyttää Internet-selaimena tai sen avulla voidaan lukea ja lähettää sähköpostia. Tulevaisuudessa myös osa julkisen hallinnon palveluista on mahdollista hoitaa digitaalisen television kautta [3].

Ilman paluukanavaa ja yhteyttä palvelun tarjoajaan voidaan toteuttaa paikallista vuorovaikutteisuutta sisältäviä palveluja. Näitä voivat olla esimerkiksi sähköinen ohjelmaopas (electronic program guide, EPG), super-tekstitelevisio sekä TV-ohjelmiin liittyvät lisäarvopalvelut. Paikallinen vuorovaikutteisuus avaa aivan uusia mahdollisuuksia myös mainostajille. Vuorovaikutteisten mainosten avulla voidaan kuluttajalle tarjota helposti monimuotoista lisämateriaalia tuotteesta tai palvelusta. Mikäli paluukanava on käytössä, katsoja voi myös tilata tuotteen TV:n kautta.

Kaupallisesti tärkeäksi palvelumuodoksi voi muodostua datan massajakelu. Digitaalista laajakaistaista TV-lähetysverkkoa voidaan käyttää esimerkiksi CD- tai DVD-levyjen sisältöjen massajakeluun. Digitaalista lähetysverkkoa voidaan käyttää myös tietokoneohjelmien jakelukanavana. [7]

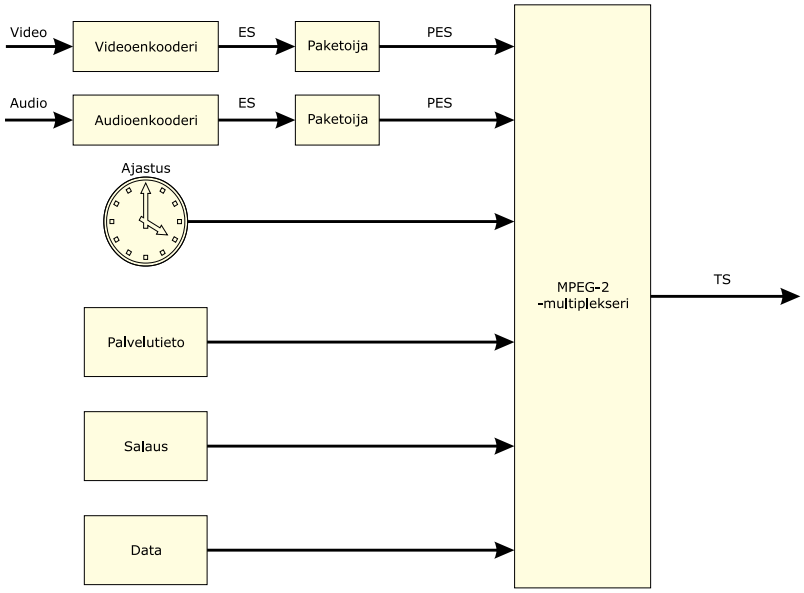
Luku 3

Datan siirto DVB-järjestelmässä

3.1 MPEG-2-virrat

MPEG-2-standardin Systems-osassa määritellään miten digitaalisessa muodossa olevat pakatut audio- ja videovirrat sekä mahdollinen muu data multipleksoidaan yhdeksi bittivirraksi, joka on sopivaa joko tallennettavaksi tai lähetettäväksi. Audio- ja videoenkooderilta tulevaa bittivirtaa nimitetään perusvirraksi (elementary stream, ES). Perusvirrasta muodostetaan paketoitu perusvirta (packetized elementary stream, PES) paketoimalla perusvirta joko vakio- tai vaihtuvakokoisiin paketteihin. Jokaiseen pakettiin liitetään otsikkokenttä, joka sisältää esimerkiksi paketin dataosuuden pituuden, prioriteetin, aikaleiman ja kyseiseltä lähteeltä saatavan datavirran nopeuden. Otsikkokentässä kerrotaan myös lähteen tunnus, joka on yksilöllinen jokaiselle lähteelle. Tunnusten avulla eri virtoihin kuuluvat PES-paketit voidaan tunnistaa. [8], [9]

Standardissa on määritelty kaksi eri tapaa multipleksoida PES-paketteja. Ohjel-



Kuva 3.1. MPEG-2-multiplekserin toimintaperiaate.

mabittivirta (program stream, PS) on tarkoitettu käytettäväksi siirtokanavissa tai tallennusmedioissa, joissa bittivirheitä tapahtuu erittäin harvoin (bit error rate, BER $< 10^{-10}$). Tällaisia ovat esimerkiksi CD- ja DVD-levyt. Siirtobittivirta (transport stream, TS) on tarkoitettu käytettäviksi virheherkillä siirtokanavilla, joissa bittivirheitä voi tapahtua suhteellisen usein (BER $> 10^{-4}$). Siirtobittivirtaa käytetään lähinnä digitaalisissa TV-lähetyksissä. Kuvassa 3.1 on esitetty MPEG-2-multiplekserin toimintaperiaate. [10]

Ohjelmabittivirta koostuu PES-paketeista ja muusta lisäinformaatiosta muodostetuista pakkauksista (packs). Pakkaus sisältää otsikon, mahdollisen järjestelmäotsikon ja vaihtelevan määrän PES-paketteja. Pakkaukseen voi sisältyä PES-paketteja monista eri perusvirroista. Pakkauksen maksimipituus on 64 kilotavua. Hitaita bittivirtoja käytettäessä on kuitenkin otettava huomioon, että paketin otsikossa kuljetetaan ajoitusinformaatiota ja tästä syystä standardissa on vaatimus, että oh-

jelmabittivirrassa on esiinnyttävä uuden paketin otsikko vähintään 0,7 sekunnin välein.

Siirtobittivirta puolestaan koostuu vakiomittaisista 188 tavun TS-paketeista (TS-packets). TS-paketti muodostuu neljän tavun otsikkokentästä, jota seuraa sovituskenttä (adaptation field) tai hyötykuorma (payload) tai molemmat. Otsikkokentän oleellisin tieto siirtobittivirran toiminnan kannalta on 13-bittinen pakettitunniste (packet identifier, PID), jonka avulla vastaanotin pystyy erottelemaan eri lähteistä tulevat paketit.

Siirtobittivirran käyttämä pieni pakettikoko mahdollistaa tehokkaiden virheenkorjausmenetelmien hyödyntämisen ja paremman virheistä toipumisen ohjelmabittivirtaan nähden. Näin ollen siirtobittivirta tarjoaa myös virheherkillä siirtoteillä tarvittavaa vikasietoisuutta. 188 tavun pakettikoko on valittu siksi, että se on sopiva ATM-verkossa käytetyn solun hyötykuorman koon (47 tavua) monikerta. Siirtobittivirrassa PES-paketit kuljetetaan TS-pakettien hyötykuormassa. Yleensä yksi PES-paketti ei mahdu TS-paketin 184 tavun hyötykuormaan, vaan se joudutaan pilkkomaan moneen eri TS-pakettiin.

Paketoinnin lisäksi siirtobittivirran toinen ero ohjelmabittivirtaan nähden on siinä, että se mahdollistaa usean toisistaan riippumattoman ohjelman välittämisen samassa siirtobittivirrassa. Ohjelmien ei tarvitse olla tahdistettuina samaan järjestelmäkelloon (system time clock, STC), vaan ne voivat olla tahdistettina eri kelloihin. Yhden ohjelman muodostavat PES-paketit on kuitenkin tahdistettava samaan kelloon.

3.2 Palvelutieto

Siirtobittivirrassa eri perusvirtoihin kuuluvat paketit erotellaan paketin otsikossa olevan PID-tunnisteen avulla. Tavallinen TV-ohjelma muodostuu yleensä kahdesta perusvirrasta, jotka kuljettavat ohjelman äänen ja kuvan. Ohjelmaan voi liittyä myös useampia perusvirtoja, jotka kuljettavat esimerkiksi eri kielisiä ääniraitoja tai ohjelmaan liittyviä lisäarvopalveluja. Jotta vastaanotin tietää millä PID-arvolla ohjelmaan kuuluvat paketit siirtobittivirrassa kulkevat, siihen on lisätty ohjaustietoa jota nimitetään palvelutiedoksi (service information, SI).

MPEG-2-standardissa määriteltyä palvelutietoa kutsutaan nimellä program specific information (PSI). PSI:n muodostaa neljä taulua: program association table (PAT), program map table (PMT), conditional access table (CAT) ja network information table (NIT). DVB on laajentanut PSI:tä määrittelemällä yhdeksän lisätaulua, jotka muodostavat DVB-järjestelmissä käytettävän DVB-SI-mekanismin [11]. Seuraavaksi käydään lyhyesti läpi PSI-taulujen merkitys.

Program association table. PAT sisältää tiedon multipleksin jokaisesta ohjelmasta. PAT:ssä on jokaista ohjelmaa kohden ohjelman tunnus ja PID, jolla ohjelmaan liittyvä PMT lähetetään. PAT lähetetään aina PID-tunnuksella 0x0000.[1]

Program map table. Jokaista siirtobittivirrassa olevaa ohjelmaa kohden lähetetään yksi PMT. PMT sisältää tiedon siitä mitä komponentteja (audio, video jne.) ohjelma sisältää ja millä PID-tunnuksella ko. komponentit lähetetään. PMT lähetetään PAT:ssä kerrotulla PID-tunnuksella.

Conditional access table. Mikäli siirtobittivirrassa on salattuja ohjelmia, CAT sisältää salauksen purkuun tarvittavat tiedot. CAT lähetetään PID-tunnuksella

[1] Merkintä 0x0000 tarkoittaa C-kielestä tuttua luvun 0 heksadesimaaliesitystä.

0x0001.

Network information table. NIT sisältää tietoa verkon fyysisestä organisoinnista sekä verkon ominaisuuksista. NIT lähetetään PAT:ssä kerrotulla PID-tunnuksella.

DVB-järjestelmissä käytettävät lisätaulut helpottavat vastaanottimen automaattista konfigurointia ja viritystä. DVB-SI sisältää myös lisätietoa multipleksissä olevista palveluista. Tätä lisätietoa hyödynnetään esimerkiksi sähköisessä ohjelmaoppaassa. DVB:n määrittelemistä yhdeksästä taulusta kolme on pakollisia ja kuusi vapaaehtoisia. Seuraavaksi esitellään DVB-SI:n pakollisten taulujen merkitys.

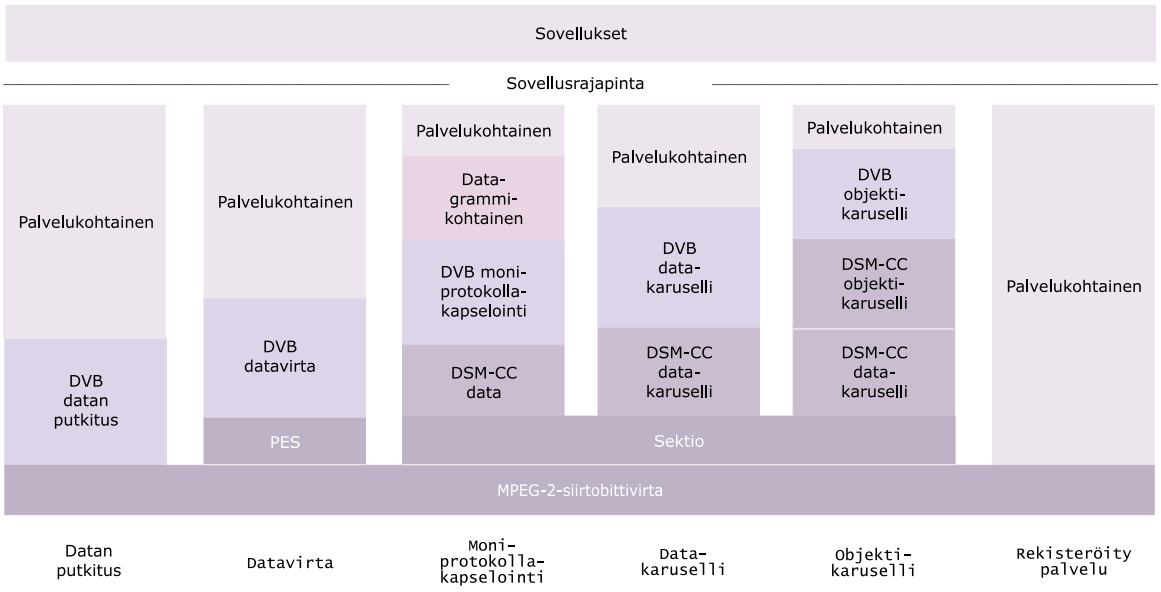
Service description table. SDT:ssä kuvataan palvelujen nimet sekä niihin liittyvät muut ominaisuudet. SDT lähetetään PID-tunnuksella 0x0011.

Event information table. EIT sisältää tietoa tällä hetkellä käynnissä olevista sekä tulevista ohjelmista. EIT lähetetään PID-tunnuksella 0x0012.

Time and date table. TDT:n avulla vastaanottimen kello ja päivämäärä voidaan päivittää automaattisesti. TDT lähetetään PID-tunnuksella 0x0014.

Vapaaehtoisia tauluja DVB-järjestelmässä ovat bouquet association table (BAT), running status table (RST), stuffing table (ST), time offset table (TOT), selection information table (SIT) ja discontinuity table (DIT).

Taulut jaetaan lähetystä varten yhteen tai useampaan sektioon, jotka sijoitetaan TS-pakettien hyötykuormaan. Sektiot määrittelevät taulujen tarkan bittitason rakenteen ja miten taulut siirretään TS-paketeissa. MPEG-2-standardissa on myös määritelty miten sektioita voidaan käyttää välittämään vapaamuotoista dataa.



Kuva 3.2. Sovellusalueet ja protokollat [13].

3.3 Datan siirtoon käytettävät protokollat

DVB-järjestelmissä datan välittämiseen siirtobittivirran mukana on monia eri tapoja, joista voidaan valita sopiva sovelluksen ja sen käyttämän datan luonteen mukaan. Kuvassa 3.2 on esitetty DVB:n määrittelemät viisi erilaista sovellusaluetta ja niihin liittyvät datan siirtoprotokollat. [12], [13]

Yksinkertaisin sovellusalueista on datan putkitus (data piping). Siirrettävä data sijoitetetaan suoraan TS-pakettien hyötykuormaan ilman lisäkapselointia. Datan putkitus ei sisällä mitään sellaisia mekanismeja, joilla sovellus saa lähetetyn datan käyttöönsä, vaan se on jätetty täysin sovelluksen tehtäväksi.

Datavirta (data streaming) tarjoaa hieman enemmän toiminnallisuutta kuin datan putkitus. Lisätoiminnallisuus koskee lähinnä parempaa ajoituksen kontrollointia. Datavirta mahdollistaa asynkronisen datan lähettämisen, synkronisen da-

tan lähettämisen (tarkat ajoitusvaatimukset) ja synkronoidun datan lähettämisen (synkronointi toisen TS:n kanssa). Datavirta perustuu PES-pakettien käyttöön.

Moniprotokollakapselointi (multi protocol encapsulation, MPE) tarjoaa keinon kuljettaa eri verkkoprotokollia siirtobittivirrassa. Se on optimoitu erityisesti IP-protokollaa silmälläpitäen, mutta LLC/NSAP-kapseloinnin avulla muidenkin protokollien siirto onnistuu. Moniprotokollakapselointi käyttää 48-bittisiä MAC-osoitteita ja tukee unicast-, multicast- sekä broadcast-lähetyksiä.

Datakarusellia (data carousel) voidaan käyttää sovelluksissa, jotka hyödyntävät datan toistuvaa lähetystä palvelimelta asiakkaille. DVB:n määrittelemässä datakarusellissa lähetettävä data on järjestetty moduuleiksi, joita palvelin lähettää toistuvasti asiakkaalle. Kun asiakassovellus haluaa käyttää jonkun moduulin dataa, sen täytyy odottaa kunnes moduuli seuraavan kerran lähetetään. Datakaruselli pohjautuu DSM-CC:n (Digital Storage Media–Command and Control) käyttöön [14]. Hyvä esimerkki datakarusellin kaltaisesta palvelusta on nykyinen teksti-TV, jossa sivuja lähetetään toistuvasti tietyn ajanjakson sisällä. NorDigin laatimissa määrittelyissä set-top boxin ohjelmiston päivitys tapahtuu datakarusellin avulla.

Objektikaruselli (object carousel) on toteutettu datakarusellin avulla ja sitä käytetään objektiryhmistä muodostuvien rakenteiden välittämiseen palvelimelta asiakkaalle siirtobittivirran mukana. Karusellissa kuljetettavat objektiryhmät koostuvat directory-, file- ja stream -objekteista. Objektikarusellin avulla voidaan siirtää palvelimella sijaitseva hakemistorakenne asiakkaalle. Palvelimella sijaisevat tiedostot ja hakemistot kuvataan objekteiksi, jotka palvelin lähettää siirtobittivirran mukana käyttäen objektikaruselliprotokollaa. DVB:n määrittelemä objektikaruselli pohjautuu DSM-CC-objektikaruselliin ja Common Object Request Broker Architecture -sovelluskehykseen [15].

Rekisteröity palvelu on jonkun tietyn organisaation varaama protokollatunniste, jolla se lähettää dataa. Varatulla protokollatunnisteella voidaan lähettää esimerkiksi organisaation omaa testidataa. [12]

Luku 4

Multimedia Home Platform

4.1 Multimedia Home Platformin tavoite

Vuonna 1997 DVB-projekti laajensi toimenkuvaansa ja rupesi kartoittamaan kaupallisia- ja teknisiä vaatimuksia ns. Multimedia Home Platformille (MHP). MHP käsittää kodin päätelaitteet (set-top box, TV, PC), tarvittavat lisälaitteet ja kotiverkon. MHP on tarkoitettu avoimeksi ja laitteistoriippumattomaksi alustaksi ja se mahdollistaa sovellusten siirrettävyyden eri päätelaitteiden välillä. MHP määrittelee rajapinnat eri laitteisto- ja ohjelmistokomponenteille, mutta ei ota kantaa niiden toteutukseen. Sovellusohjelmointirajapinnan (application programming interface, API) määrittely on yksi MHP:n määrittelyn keskeisiä tavoitteita.

MHP:n määrittely on parhaillaan käynnissä ja sen odotetaan valmistuvan vuoden 2000 alkupuolella. Vaikka määrittely on vielä kesken ja ratkaistavana on monia avoimia kohtia, määrittelyn luonnos on jo varsin käyttökelpoinen vastaanotinlaitteistojen valmistajien suunnittelutyöhön.

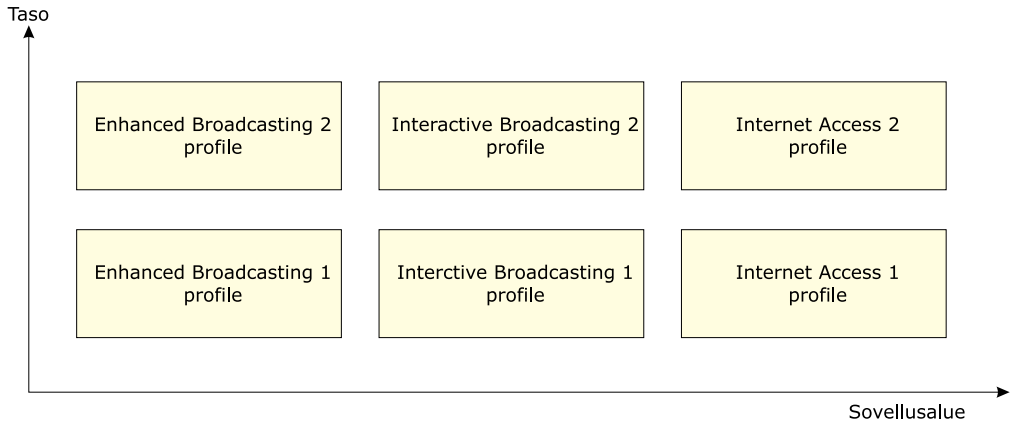
Tämä luku perustuu Multimedia Home Platformin määrittelyn luonnokseen [16] sekä artikkeleihin [17], [18] ja [19].

4.2 Sovellusprofiilit

MHP:n kannalta käytettävät sovellukset voidaan jakaa kolmeen eri sovellusalueeseen: enhanced broadcastingiin, interactive brodcastingiin ja Internet accessiin. Enhanced broadcasting -sovellusalue yhdistää normaalit audio- ja videopalvelut sekä ladattavat sovellukset, joiden avulla voidaan toteuttaa paikallista vuorovaikutusta sisältäviä palveluita. Tämän tyyppiset palvelut eivät tarvitse paluukanavaa. Interactive broadcasting mahdollistaa vuorovaikutteiset palvelut, jotka ovat joko itsenäisiä palveluja tai liittyvät lähetettävään TV-ohjelmaan. Nämä palvelut vaativat paluukanavan toimiakseen. Internet access -sovellusalue on tarkoitettu Internet-palvelujen käyttöön. Sovellusalue käsittää myös Internet-palvelujen ja broadcast-palvelujen väliset linkit.

DVB on laatinut vastaanottimille eri tasoisia profiileita, joita on sovitettu edellä kuvatuille kolmelle sovellusalueelle. Profiilien ja tasojen avulla kuvataan ominaisuuksia, jotka sovellus vaatii toimiakseen ja toisaalta ominaisuuksia, jotka vastaanotinlaitteisto tarjoaa. Kuvassa 4.1 on esitetty eri tasoisten profiilien ja sovellusalueiden suhde.

Eri tasoisilla profiileilla pyritään ottamaan huomioon sovellusten ja vastaanottimien kehitys ajan kuluessa. Profiileihin voidaan lisätä uusia tasoja jos vastaanotinlaitteiston tehovaatimukset tai muut ominaisuudet ovat kasvaneet. Profiilit mahdollistavat eri ominaisuuksilla varustettujen vastaanottimien valmistuksen. Mikäli kuluttaja ei ole kiinnostunut vuorovaikutteisista palveluista, hän voi hank-



Kuva 4.1. DVB:n määrittelemät tasot ja profiilit.

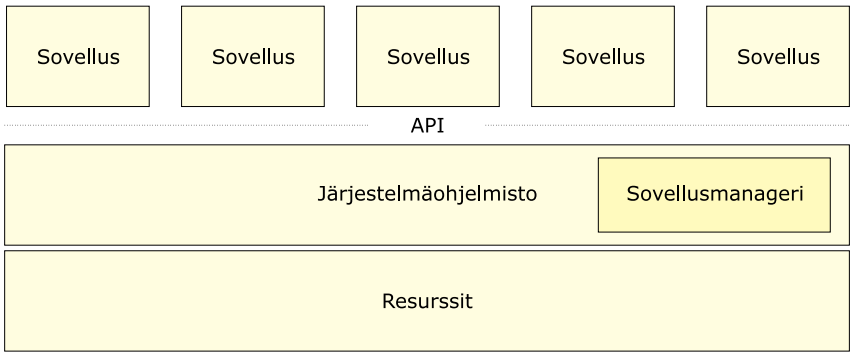
kia halvemman enhanced broadcasting -profiilin mukaisen vastaanottimen. Samoin kuin tasoja, uusia sovellusalueita sisältäviä profiileja voidaan lisätä tarpeen vaatiessa.

MHP-spesifikaation ensimmäinen versio sisältää määrittelyt enhanced broadcasting ja interactive broadcasting -profiileille. Internet access -profiilin määrittely sisällytetään spesifikaation toiseen versioon.

4.3 Arkkitehtuuri

MHP:n ohjelmiston perusarkkitehtuuri mallinnetaan kuvan 4.2 mukaisesti kolmella eri kerroksella, jossa sovellusohjelmointirajapinta sijaitsee sovelluskerroksen ja järjestelmäohjelmistokerroksen välissä.

MHP:n laitteistokomponentit esitetään laitteisto- ja ohjelmistoresursseina, joiden ryhmittelyyn määrittely ei ota kantaa. Mallissa oletetaan, että alusta voi sisältää tiettyä laitteistokomponenttia enemmän kuin yhden kappaleen. Abstrahoinnin



Kuva 4.2. MHP:n arkkitehtuurin kolme kerrosta [16].

näkökulmasta katsottuna ei ole merkitystä, kuvataanko loogiset resurssit yhdeksi vai useammaksi laitteistokomponentiksi, kunhan alusta tarjoaa resurssit sovelluksille läpinäkyvästi. Sovellusten pitää pystyä operoimaan paikallisia resursseja ikään kuin ne olisivat yhden komponentin elementtejä.

Järjestelmäohjelmisto eristää sovellukset laitteistosta ja mahdollistaa sovellusten siirrettävyyden. Sovellukset eivät käsittele resursseja suoraan, vaan järjestelmäohjelmiston tarjoaman standardoidun abstraktin rajapinnan kautta. Sovellusmanageri kuuluu tärkeänä osana järjestelmäohjelmistoon. Sen tehtävänä on kontrolloida sovellusten elinkaarta (life cycle). Elinkaaren hallintaan kuuluu sovellusten käynnistäminen, lopettaminen, järjestelmästä poistaminen sekä sovellusten käyttämän muistin hallinta.

Sovellukset ovat laitteistoriippumattomia ohjelmistokomponentteja, joiden avulla toteutetaan käyttäjälle tarjottavat palvelut. Ne toteutetaan käyttämällä järjestelmäohjelmiston tarjoamia rajapintoja.

4.4 DVB-J-alusta

Tietotekniikan alalla tuskin mikään on saanut niin paljon julkisuutta niin nopeasti kuin Sun Microsystemsin kehittämä Java. Javasta ei ainoastaan kirjoiteta tietokonelehdissä alan ammattilaisille, vaan se on löytänyt tiensä myös sanoma- ja aikakauslehtien palstoille. Java-ohjelmat ovat tulleet suurelle yleisölle tutuksi lähinnä Internetin web-sivuille lisättyjen applettien muodossa. Alunperin Java kuitenkin kehitettiin ohjelmointiympäristöksi sulautettuihin järjestelmiin. Nyt kun suurin kohu Javan ympärillä alkaa tasaantua, se on löytänyt tiensä web-sivuilta takaisin alkuperäiseen kohteeseensa, sulautettuihin järjestelmiin. [20], [21]

Vaikka Java ei sinänsä tarjoa mitään uutta ja mullistavaa tekniikkaa, siitä on onnistuttu samaan varsin toimiva kokonaisuus keräämällä yhteen nippu vanhoja hyväksi todettuja tekniikoita. Java-tekniikka onkin saanut yritysmaailmassa taakseen melkoisen tuen — niin teknisessä kuin taloudellisessakin mielessä. Pitkällisen harkinnan ja kädenväännön jälkeen myös DVB on valinnut Javan käytettäväksi Multimedia Home Platformissa.

Yksi tärkeimpiä ominaisuuksia mitä Java tarjoaa on laitteistoriippumattomuus. Java-ohjelmat eivät ole sidottuja tiettyyn prosessoriarkkitehtuuriin tai käyttöjärjestelmään, vaan ne ovat siirrettävissä toisiin laitteistoihin ilman muutoksia. Tämä on saatu aikaan käyttämällä virtuaalikonetta. Käännettäessä Java-ohjelman lähdekoodi Java-kääntäjällä lopputuloksena on laitteistoriippumatonta tavukoodia (bytecode). Käännetty tavukoodi suoritetaan virtuaalikoneessa (virtual machine, VM), jota voidaan ajatella eräänlaisena matalan tason tulkkina. Java-ohjelmia voidaan siis suorittaa kaikissa ympäristöissä, joihin on saatavana Java-virtuaalikone ja tarvittavat Java-kirjastot. Laitteistoriippumattomuudesta ja sovellusten siirrettävyydestä juontaakin juurensa Javan kuuluisa markkinointilause “write

once, run everywhere". [20]

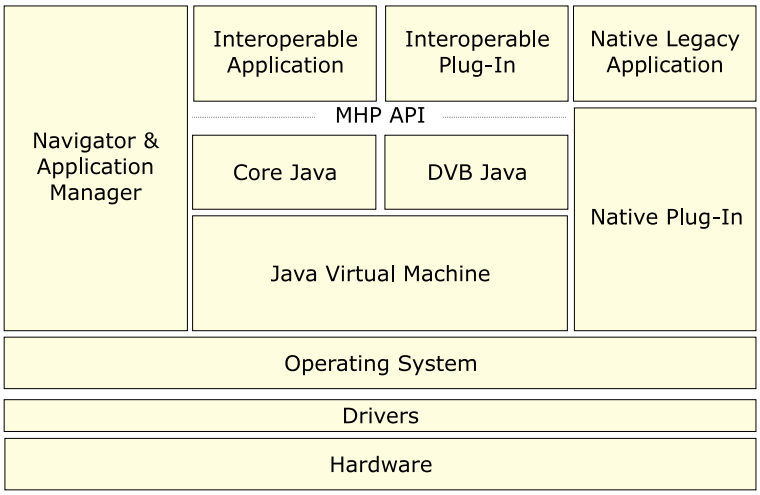
MHP:ssä käytettävä Java-alusta, josta käytetään nimitystä DVB-Java tai DVB-J, perustuu Java-spesifikaatioiden mukaisen virtuaalikoneen [22] ja kirjastojen [23] käyttöön. Muistin säästämiseksi kirjastoista on kuitenkin poistettu ominaisuuksia, joita ei ole katsottu tarpeellisiksi TV-ympäristössä. MHP sisältää Javan peruskirjastojen lisäksi laajan joukon ohjelmistorajapintoja, jotka on tarkoitettu TV-spesifisten toimintojen toteuttamiseen. Monet näistä rajapinnoista ovat peräisin Digital Audio-Visual Council (DAVIC) -organisaation tekemistä määrittelyistä [24].

MHP:hen sisältyvät TV-spesifiset rajapintamäärittelyt voidaan jaotella karkeasti neljään pääluokkaan, joita kuvataan seuraavaksi lyhyesti.

Presentation APIs. Graafinen käyttöliittymä pohjautuu Javan Abstract Window Toolkittiin (AWT). AWT:stä ovat mukana lightweight-komponenttien toteuttamiseen tarvittavat luokat. Varsinaiset käyttöliittymäkomponentit toteutetaan lightweight-komponentteina, jotka perustuvat HAVi-spesifikaation [25] mukaisiin komponentteihin. Audion ja videon esittämiseen käytetään Java Media Frameworkiä [26].

Data access APIs. Tähän ryhmään kuuluvat rajapinnat liittyvät siirtobittivirrassa kuljettetavan datan käsittelyyn. Mukana ovat mm. rajapinnat, joilla voidaan käsitellä MPEG-2-sektioita (section filtering API) sekä rajapinnat data- ja objektikarusellien hallintaan.

Service information and selection APIs. Tämä ryhmä sisältää palvelutietojen ja palvelujen hallintaan liittyviä rajapintoja. Sisältää myös virittimen ohjaukseen liittyvän rajapinnan (tuning API) sekä salaukseen ja valtuutukseen liittyvät rajapinnat (conditional access APIs).



Kuva 4.3. MHP-vastaanottimen ohjelmiston kerrosmalli.

I/O device APIs. Tähän ryhmään kuuluvat ulkoisten liityntöjen ohjaamiseen liittyvät rajapinnat. Mukana ovat sarja- ja rinnakkaisporttien sekä toimikorttien käsittelyyn liittyvät rajapinnat.

Edellisten lisäksi spesifikaation luonnoksessa mainitaan useita vielä määrittelemättömiä rajapintoja.

Käytännön vastaanotinlaitteistoissa ohjelmisto muodostuu kuvan 4.3 mukaisesta kerrosmallista. Järjestelmän ytimenä on reaaliaikakäyttöjärjestelmä ja tarvittavat laiteohjaimet (drivers). Käyttöjärjestelmän päällä on Java-sovituskerros, joka tarjoaa laitteistoriippumattoman Java-rajapinnan (MHP API) sovellusten käyttöön. Java-sovituskerros koostuu Java-virtuaalikoneesta, Java-standardikirjastoista sekä DVB-J-kirjastoista. Oleellisena osana järjestelmäohjelmistoon kuuluvat sovellusmanageri sekä navigaattori. Navigaattori on yksinkertainen käyttöliittymä, jonka avulla käyttäjä voi ohjata vastaanotinta.

Koska DVB:ssä katsottiin tarpeelliseksi, että MHP-vastaanottimessa voitaisiin

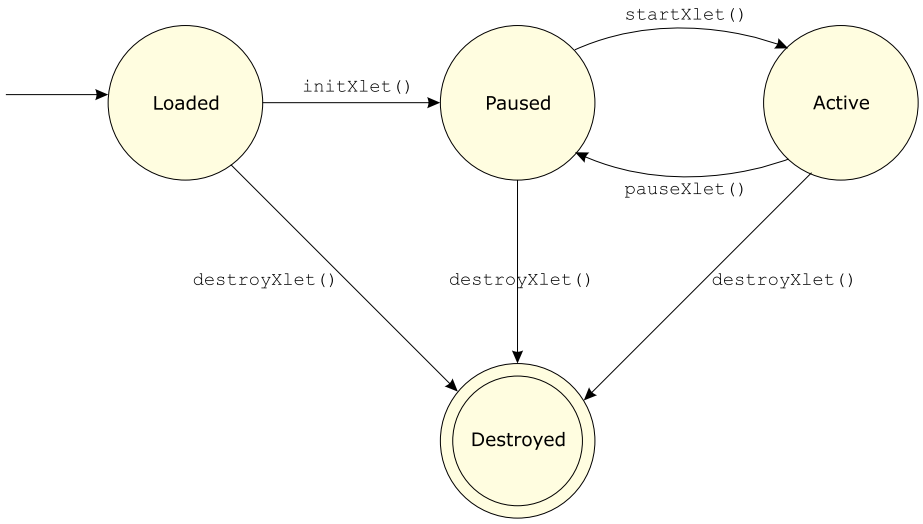
käyttää jo olemassa oleville järjestelmille (OpenTV, MediaHighway, Power TV jne.) tehtyjä sovelluksia, siihen sisällytettiin ns. Plug-In-mekanismi, jonka avulla vastaanottimessa voidaan suorittaa myös muita kuin Java-sovelluksia. Plug-Inin tarkoituksena on tarjota MHP:n ulkopuoliselle sovellukselle sen tarvitsema rajapinta. Plug-Inin toteuttamiseen on kaksi vaihtoehtoista tapaa. Native Plug-In on toteuttu suoraan käyttöjärjestelmän päälle, eikä sen toteutus hyödynnä MHP API:a. Interoperable Plug-In on tavallaan tavallinen sovellus, joka muodostaa sovituskerroksen ulkopuolisen sovelluksen ja MHP API:n välille. Interoperable Plug-In on toteutettu käyttämällä ainoastaan MHP API:a.

4.5 DVB-J-sovellukset

DVB-J-alustan päällä ajettavat DVB-J-sovellukset muodostuvat Java-luokista, jotka ovat käännetty Java-kääntäjällä tavukoodiksi. DVB-J-sovellukset muistuttavat jossain määrin webistä tuttuja appletteja, minkä takia niitä kutsutaan myös nimellä Xlet. Xlet-sovelusten semantiikka on kuitenkin määritelty huomattavasti tarkemmin kuin applettien.

Xlet-sovelluksen elinkaarimallissa määritellään protokolla (dialogi) Xletin ja ympäristön välillä. Jotta elinkaarimalli voitaisiin määritellä selkeästi, sen tulee sisältää seuraavat ominaisuudet:

- tilakone, joka on yksinkertainen ja hyvin määritelty
- lyhyesti ja ytimekkäästi määritellyt tilat
- API, jolla voidaan signaloida tilojen vaihdokset.



Kuva 4.4. Xletin tilakonemalli [16].

Television katsojilla on selkeä käsitys siitä, miten television tulee toimia. Vuorovaikutteisten Xlet-sovellusten tulee käyttäytyä mahdollisimman pitkälle niiden odotusten mukaan, mitä television katsojat käyttämilleen palveluille asettavat. Television katsojat ovat esimerkiksi tottuneet siihen, että valitut toiminnot tapahtuvat heti kun valinta on tapahtunut, joten Xlet-sovellusten käynnistyksestä aiheutuva latenssi ei saa olla kovin suuri. Xlet on myös voitava keskeyttää ja poistaa käytöstä milloin tahansa. Kuvassa 4.4 on esitetty Xlet-sovelluksen tilakoneen malli, joka on suunniteltu niin, että se toteuttaa edellä kuvatut vaatimukset.

Xlet voi vaihtaa tilaa joko sovellusmangerin toimesta tai sovellus voi itsenäisesti mennä tilasta toiseen. Mikäli sovellus vaihtaa tilaa itsenäisesti, sen on signaloitava sovellusmanageria tästä. Taulukossa 4.1 on selvitys Xletin tilasiirtymistä ja eri tilojen merkityksestä.

Kommunikointi Xletin ja sen ympäristön välillä tapahtuu Xlet API:n kautta. API:n avulla sovellusmanageri voi siirtää Xletin toiseen tilaan tai Xlet voi ilmoittaa ta-

Taulukko 4.1. Kuvaus Xlet-tilakoneen tiloista ja siirtymistä

Tila	Kuvaus
Loaded	Xlet on ladattu mutta ei alustettu. Tilaan tullaan seuraavassa tapauksessa: • Uusi Xlet-instanssi on luotu `new`-operaattorilla. Tyypillisesti tässä vaiheessa Xlet tekee pieniä alustustoimenpiteitä. Poikkeuksen tapahtuessa Xlet menee Destroyed-tilaan.
Paused	Xlet on alustettu ja valmis käynnistymään. Xletin ei pitäisi varata tai käyttää jaettuja resursseja tässä tilassa. Tilaan tullaan seuraavissa tapauksissa: • Loaded-tilasta sen jälkeen, kun suoritus on palannut onnistuneesti suoritetusta `Xlet.initXlet()`-metodista. • Active-tilasta sen jälkeen, kun suoritus on palannut onnistuneesti suoritetusta `Xlet.pauseXlet()`-metodista. • Active-tilasta ennen suorituksen palaamista Xletiin onnistuneesti suoritetun `XletManager.paused()`-metodin jälkeen.
Active	Xlet on suorituksessa normaalisti ja sen tarjoama palvelu on käytössä. Tilaan tullaan seuraavassa tapauksessa: • Paused-tilasta sen jälkeen, kun suoritus on palannut onnistuneesti suoritetusta `Xlet.startXlet()`-metodista.
Destroyed	Xlet on vapauttanut varaamansa resurssit ja lopettanut toimintansa. Tilaan tullaan seuraavissa tapauksissa: • Suoritus on palannut onnistuneesti suoritetusta `Xlet.destroyXlet()`-metodista. • Suoritus on palannut Xletiin onnistuneesti suoritetusta `XletManager.destroyed()`-metodista.

pahtunnesta tilasiirroksesta sovellusmanagerille. Xlet API koostuu neljästä metodista:

`public void initXlet(XletManager ctx)` Alustaa Xletin. Tämä metodi signaloi Xletille, että sen pitää suorittaa tarvittavat alustustoimenpiteen, jotta se pystyy aloittamaan toimintansa kohtuullisessa ajassa. Metodi saa parametrina `XletManager`-objektin, jonka kautta Xlet pääsee käsiksi ympäristön resursseihin. `XletManager`-objektin avulla Xlet myös pystyy signaloimaan sovellus-

managerille tilamuutoksista.

`public void startXlet()` Tämän metodin suorituksen jälkeen Xlet on Active-tilassa ja sen tarjoama palvelu toimii normaalisti.

`public void pauseXlet()` Tällä metodilla signaloidaan Xletille, että sen sen tulee pysäyttää tarjoamansa palvelu ja mahdollisuuksien mukaan vapauttaa käyttämänsä jaetut resurssit. Metodin suorituksen jälkeen Xlet on Paused-tilassa.

`public void destroy()` Xletin tarjoamaa palvelua ei enää tarvita. Xletin täytyy suorittaa tarvittavat lopputoimenpiteet ja vapauttaa varaamansa resurssit.

Mikäli Xlet ei virhetilanteen tai jonkin muun syyn takia pysty siirtymään haluttuun tilaan, se viestittää tästä sovellusmanagerille `XletStateChangeException`-poikkeuksen avulla.

4.6 Sovellusten siirtäminen siirtobittivirrassa

Sovellusten siirtämistä siirtobittivirrassa varten MHP-spesifikaatiossa on määritelty kaksi uutta DVB-SI-taulua. Uusia tauluja käytetään sovelluksia kuvaavien tietojen välittämiseen. Seuraavaksi kuvaillaan tauluja lyhyesti.

Application information table. AIT:ssä on tiedot, jotka vastaanotin tarvitsee, jotta sovellus voidaan ladata ja käynnistää. Näitä tietoja ovat mm. profiili, jonka mukaisessa laitteessa sovellus on tarkoitettu suoritettavaksi, sovelluksen nimi, sovelluksen siirtämiseen käytetty protokolla (objektikaruselli, MPE jne.) ja tarvittavat käynnistysparametrit.

Version summary table. VST sisältää tiivistetyssä muodossa AIT:ssä olevat tiedot. Koska AIT:n toistointervalli saattaa olla hyvinkin pitkä ja sen koko suuri, käytetään VST:tä nopeana tiedonvälittäjänä kertomaan multipeksissä olevista sovelluksista ja niitä koskevista muutoksista.

Yleensä sovellukset välitetään objektikarusellissa, mutta muidenkin protokollien käyttö on mahdollista.

Luku 5

DVB-vastaanotinkortin Java-rajapinta

5.1 Yleistä

Future TV on TEKES:n ja yritysten rahoittama kansallinen hanke, jonka tavoitteena on mm. tutkia digitaalisen television uusia palvelumuotoja. Tätä varten Tampereen teknillisen korkeakoulun Digitaalisen median instituutissa kehitellään PC-ympäristöön yleiskäyttöistä alustaa, jossa näiden uusien palvelujen käytännön testaaminen onnistuu. Erityisen mielenkiinnon kohteena on Multimedia Home Platformin mukaisten, Javalla toteutettujen palvelusovellusten testaus ja niiden välittäminen digitaalisen TV-lähetteen mukana.

Digitaalisen TV-lähetteen vastaanotto PC:llä tapahtuu sopivan lisäkortin avulla. Tämän työn käytännön osuuden tarkoituksena oli implementoida Java-rajapinta DVB-yhteensopivaan vastaanotinkorttiin Windows 98 -ympäristössä. Rajapinta mahdollistaa vastaanotinkortin konfiguroinnin ja MPEG-2-virtojen vastaanottamisen Java-ohjelmasta käsin. Työssä käytettiin tanskalaisen COCOM:n valmistamaa vastaanotinkorttia, joka on yhteensopiva DVB-S-standardin kanssa. Vastaanotinkortin tehtävänä on purkaa vastaanotetun lähetteen kanavakoodaus ja erot-

taa siirtobittivirrasta halutun PID-tunnisteen sisältävät paketit, jotka kortti edelleen välittää väylän kautta PC:n muistiin jatkokäsittelyä varten. Työssä käytetty kortti sisältää ainoastaan vastaanottoon tarvittavat lohkot, joten TV-kuvan esittämiseen tarvitaan lisäksi MPEG-2-purkukortti tai kuvan ja äänen purkaminen on tehtävä ohjelmallisesti.

5.2 Suunnittelu- ja toteutusnäkökohtia

Ohjelmiston suunnittelun ja toteutuksen lähtökohdaksi otettiin uudelleenkäytettävyys. Java-rajapinnan muodostavat pakkaukset, luokat ja metodit muodostavat kirjaston, jonka tulee sopia käytettäväksi monenlaisten ohjelmistojen rajapintana vastaanotinkorttiin ilman, että kirjaston lähdekoodia tarvitsee muokata. Kirjaston tulee lisäksi tarjota yleisimmät datan vastaanottoon liittyvät palvelut, jotta niitä ei erikseen tarvitse rakentaa joka sovellukseen. Edellä mainituista seikoista johtuen, kirjaston suunnittelussa ja toteutuksessa kiinnitettiin erityistä huomiota seuraaviin seikkoihin:

1. Kirjaston pitää olla helposti laajennettavissa. Kirjasto muodostaa perustan Future TV -projektissa toteutettavalle uusien televisiopalvelujen testausympäristölle, joten sitä pitää pystyä käyttämään monien erilaisten ohjelmien yhteydessä joilla kuitenkin on paljon yhtäläisyyksiä.

2. Uusien palvelujen toteutukset perustuvat yleensä palvelutiedon sekä data- ja objektikarusellien käyttöön, jotka välitetään siirtobittivirrassa sektioina. Yleensä sektioiden suodatus tapahtuu laitteistolla, mutta koska COCOM:n kortti ei sisällä tätä toiminnallisuutta, kirjaston tulee tarjota ohjelmallisesti toteutettu sektioiden suodatus.

3. Vastaanotinkortin C-kielinen rajapinta on hankala ja työläs käyttää ja sillä ohjelmoitaessa tekee helposti virheitä. Java-rajapinnan käytön tulee olla helposti opittavissa ja sillä tulee olla Javan standardikirjastojen mukainen “look & feel”.

4. Kirjaston toteutuksessa käytetään natiivikoodia, joten siitä tulee laitteistoriippuva. Kirjastosta tehdään mahdollisesti tulevaisuudessa Linux-ympäristössä toimiva versio, joten laitteistoriippuvan koodin määrä tulee minimoida ja käytetyn koodin on oltava mahdollisimman hyvin siirrettävää lähdekooditasolla. Toisaalta kirjaston avulla käsitellään suurinopeuksisia bittivirtoja mistä seuraa, että kirjastolle on asetettu suuret tehokkuusvaatimukset, jolloin natiivikoodin käyttö voi olla perusteltua. Siirrettävyyden ja tehokkuuden välille pitää löytää sopiva tasapaino.

5. Rajapinnan dokumentointi generoidaan Javan käytännön mukaan sopivalla työkalulla (javadoc) lähdekoodiin sisältyvistä kommenteista. Rajapinnan dokumentoinnin lisäksi kirjaston käytöstä täytyy olla tarpeeksi esimerkkiohjelmia.

Lisäksi ohjelmiston koodin ja dokumentoinnin pitää olla ohjelmistoalalla yleisesti käytettyjen tapojen mukaisesti tehty, millä varmistetaan sen helppo ylläpidettävyys.

5.3 Vastaanotinkortin ohjelmistorajapinta

Kortin mukana toimitetaan Windows 98 -ympäristöön ohjelmistorajapinta, jonka kautta kortin ohjaus tapahtuu. Rajapinta käsittää muutaman C-kielisen otsikkotiedoston, joissa määritellään kortin kanssa kommunikointiin tarvittavia vakioi-

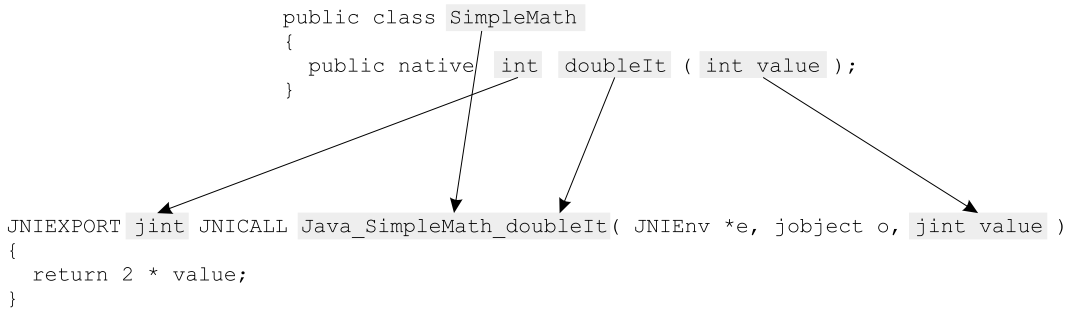
ta ja tietorakenteita. Kortin ohjaus sovelluksesta tapahtuu Win32 API:hin kuuluvan `DeviceIoControl()`-funktion avulla. Funktiolle annetaan parametriksi osoitin tietorakenteeseen, jonka välityksellä sovellusohjelma ja kortin laitteistoajuri kommunikoivat. Sovellusohjelma tallettaa tietorakenteeseen suoritettavan ohjaustoiminnon koodin sekä tähän mahdollisesti liittyvät parametrit ja kutsuu em. funktiota. Laiteohjain puolestaan suorittaa halutun ohjaustoiminnon ja tallettaa mahdolliset paluuarvot sekä status-koodin tietorakenteeseen, josta sovellus saa ne käyttöönsä.

Rajapinnan primitiivisyydestä johtuen sen käyttö on hankalaa. Toisaalta sen käyttämiseen ei tarvita mitään ulkoisia kirjastoja, mikä mahdollistaa periaatteessa minkä tahansa ohjelmointikielen käyttämisen kortin ohjaamiseen. Edellytyksenä on, että käytettävä kieli mahdollistaa sopivien tietorakenteiden määrittelyn ja että siitä voidaan kutsua Win32 API:n `DeviceIoControl()`-funktiota.

5.4 Ulkoisten aliohjelmien liittäminen Javaan

Java Native Interface (JNI) [27] on Java Development Kitiin (JDK) kuuluva rajapinta, jonka kautta virtuaalikoneessa suoritettava Java-koodi voi käyttää toisilla ohjelmointikielillä kirjoitettuja ohjelmia ja kirjastoja. JNI:n käyttö voi olla tarpeellista esimerkiksi silloin, kun halutaan tehdä liityntöjä laitteistoon tai käyttää jo olemassa olevia kirjastoja.

JNI:n avulla natiivikoodi voi käsitellä Java-objekteja samaan tapaan kuin Java-koodi niitä käsittelee. Natiivikoodi voi käyttää Java-koodissa luotuja objekteja tai se voi luoda uusia objekteja. Luodut objektit voidaan välitää Java-koodille, joka käsittelee niitä aivan tavallisina Java-objekteina.



Kuva 5.1. Natiivimetodin esittely ja toteutus.

Natiivikoodista voidaan kutsua helposti Java-koodissa olevia metodeja ja välittää niille halutut parametrit. Natiivikoodissa voidaan myös käsitellä poikkeuksia sekä luoda uusia poikkeuksia, jotka käsitellään Java-koodissa. JNI mahdollistaa myös ajoaikaisten tyyppitarkistusten tekemisen.

Natiivimetodin liittäminen luokkaan tapahtuu lisäämällä määre `native` metodin esittelyyn. Koska metodille ei määritellä toteutusta, esittely päättyy poikkeuksellisesti puolipisteeseen. Kuvassa 5.1 on luokan `SimpleMath` määrittely, joka sisältää natiivimetodin `doubleIt` esittelyn. Kuvassa on myös esitetty natiivimetodin toteutus C-kielellä.

Natiivimetodin nimi saadaan yhdistämällä Java-etuliite, luokan nimi ja metodin nimi. Mikäli luokka kuuluu johonkin muuhun kuin oletuspakkaukseen, lisätään luokan nimen eteen vielä pakkauksen nimi. Natiivimetodille välitetään kaksi ylimääräistä parametria metodin normaalien parametrien lisäksi. `JNIEnv`-tyyppisen parametrin avulla natiivimetodi pääsee käsiksi Java-ympäristöön esimerkiksi Javalla toteutettujen metodien kutsumista varten. `jobject`-tyyppinen parametri on osoitin siihen olioon, jonka kautta natiivimetodia kutsuttiin. Se vastaa C++-kielen `this`-osoitinta. JDK:n mukana tulee javah-työkalu jolla C- ja C++-kielisten funktioesittelyjen generoiminen Java-luokkatiedostosta onnistuu kätevästi.

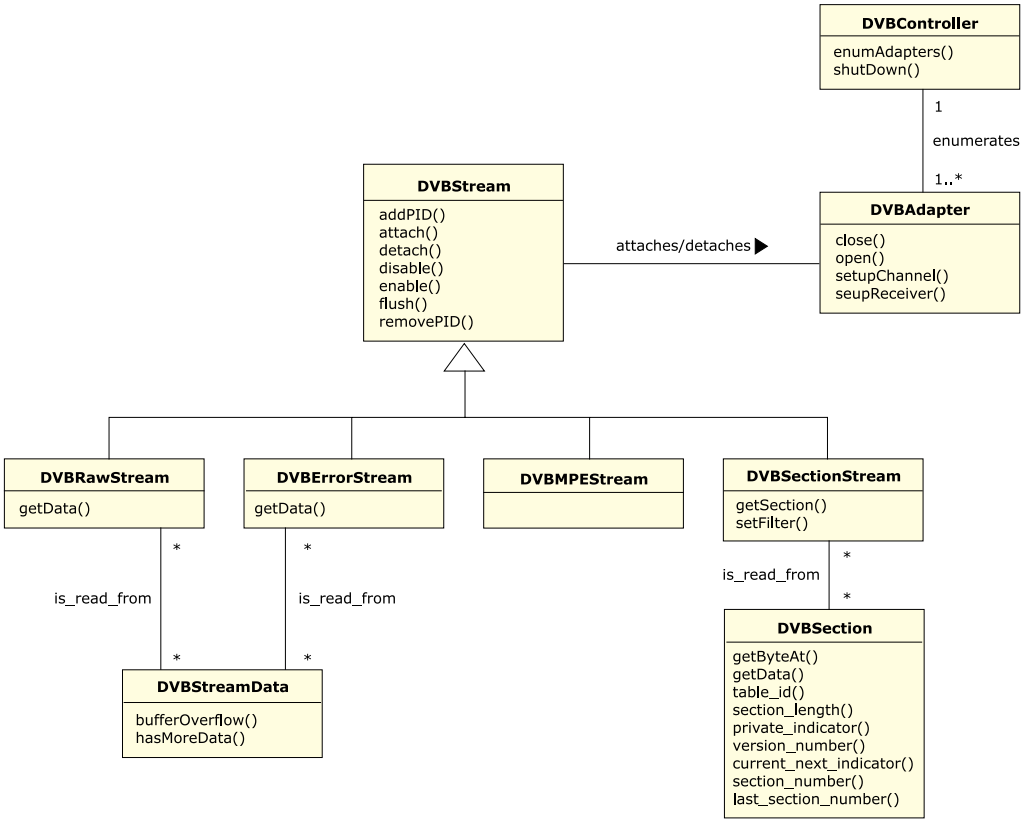
Natiivimetodien toteutukset käännetään ajoympäristökohtaisiksi jaetuksi kirjastoksi, joka ladataan Java-ohjelmassa `System.loadLibrary()`-metodilla. Mikäli käytettävä ympäristö ei tue dynaamisesti ladattavia jaettuja kirjastoja, pitää natiivimetodien koodi linkittää stattisesti virtuaalikoneeseen.

5.5 Arkkitehtuuri

Koska tarkoituksena oli tehdä ohjelmisto, jonka rajapinnat olisivat mahdollisimman paljon Javan standardikirjastojen mukaisia, korttiin liittyvät loogiset kokonaisuudet ja niihin liittyvät ohjaustoiminnot oli luonnollista mallintaa luokiksi ja niihin kuuluviksi operaatioiksi. Kuvassa 5.2 on esitetty UML-notaatiolla ohjelmiston tärkeimmät luokat ja niiden suhteet. Koko ohjelmisto koostuu 26 luokasta, jotka sisältävät yhteensä 111 metodia.

`DVBController`-luokka tarjoaa palvelut, joiden avulla päästään käsiksi järjestelmään asennetuihin vastaanotinkortteihin. Luokka sisältää vain staattisia metodeja, joten siitä ei tehdä instansseja. Metodi `enumAdapters()` palauttaa `DVBAdapter`-tyyppiä olevan taulukon. Taulukossa olevat objektit mallintavat kukin yhtä järjestelmään asennettua vastaanotinkorttia. Vastaanotinkortin laiteohjain ei vielä tue useamman kuin yhden kortin käyttöä, joten `enumAdapters()`-metodi voisi palauttaa pelkän `DVBAdapter`-objektin. Taulukon käytöllä on kuitenkin mahdollista lisätä monen kortin tuki jälkeenpäin muuttamatta luokan julkista rajapintaa. Luokan staattiseen alustuslohkoon (static initialization block) on sijoitettu natiivikoodin sisältävän dynaamisen kirjaston lataus, joten natiivikoodi latautuu automaattisesti virtuaalikoneen ottaessa luokan käyttöön, eikä käyttäjän tarvitse huolehtia sen lataamisesta.

`DVBAdapter`-luokka mallintaa vastaanotinkorttia ja se sisältää metodit, joilla



Kuva 5.2. Java-rajapinnan yksikertaistettu luokkakaavio.

vastaanotinkortille asetetaan viritys-, virheenkorjaus- ja vastaanottoparametrit. Luokka sisältää myös metodit, joilla asetetut parametrit voidaan lukea kortilta.

`DVBStream`-luokka on abstrakti kantaluokka, joka mallintaa MPEG-2-virtoja. Metodilla `addPID()` asetetaan vastaanotettavan MPEG-2-virran PID-tunniste. Vastaanotinkortti mahdollistaa useamman kuin yhden PID-tunnisteen asettamisen, jolloin sovellus voi helposti lukea samasta `DVBStream`-objektista sekä audio- että videopaketteja. `DVBStream`-luokasta on periytetty neljä luokkaa, joiden kautta eri tyyppisen datan vastaanotto tapahtuu. Ennen kuin datan vastaanotto voi alkaa, on `DVBStream`-objektin kytkeydyttävä vastaanotinkorttia mallintavaan `DVBAdapter`-objektiin `attach()`-metodilla.

`DVBRawStream`-luokka on tarkoitettu normaalien MPEG-2-virtojen vastaanottoon. Sen kautta voidaan lukea 188 tavun mittaisia TS-paketteja. Datan lukeminen tapahtuu `getData()`-metodilla.

`DVBErrorStream`-luokka vastaa toiminnaltaan `DVBRawStream`-luokkaa, mutta siitä voidaan lukea paketit jotka kortti on vastaanottanut virheellisesti. Datan lukeminen tapahtuu `getData()`-metodilla.

`DVBMPEStream`-luokkaa käytetään vastaanottamaan MPEG-2-virran mukana moniprotokollakapseloinnin avulla kuljetettavaa dataa. Tällä luokalla ei ole muiden `DVBStream`-luokasta perittyjen luokkien tapaan `getData()`-metodia, vaan vastaanotinkortti ohjaa vastaanotetun datan käyttöjärjestelmälle ja se voidaan lukea kullekin protokollalle ominaisen rajapinnan kautta. Yleisimmin moniprotokollakapseloinnin avulla välitetty protokolla on IP, jolloin vastaanotettu data voidaan lukea normaalin socket-rajapinnan kautta.

`DVBSectionStream`-luokka on tarkoitettu MPEG-2-virrassa kuljetettavien sektioiden vastaanottoon. Usein on tarpeellista, että sovelluksella on mahdollisuus asettaa erilaisia kriteerejä sille, minkä tyyppisiä sektioita halutaan vastaanottaa. Haluttujen kriteerien täyttävien sektioiden poimintaa MPEG-2-virrasta nimitetään sektioiden suodattamiseksi (section filtering). Halutut kriteerit puolestaan määritellään suodattimilla (filters). `DVBSectionStream`-luokalla on kuormitettu metodi `setFilter()`, jolla voidaan asettaa halutun tyyppinen suodatin. Sektioiden lukeminen tapahtuu `getSection()`-metodilla, joka palauttaa `DVBSection`-objektin. Sektioiden suodatus suoritetaan yleensä laitteistolla, mutta koska työssä käytetty vastaanotinkortti ei tarjoa tätä toiminnallisuutta, on sektioiden suodatus toteutettu ohjelmallisesti. Sektioiden suodatuksen periaatteita käsitellään tarkemmin kohdassa 5.6.

`DVBSection`-luokka mallintaa MPEG-2-virrasta suodatettua sektiota. Luokka perustuu DAVIC:n määrittelemään `Section`-luokkaan[1] ja se on suunniteltu niin, että DAVIC:n määrittelemä luokka voidaan toteuttaa helposti periyttämällä ja lisäämällä muutaman puuttuvan metodin toteutus. Luokka sisältää metodit, joilla päästään helposti käsiksi sektion otsikkotietoihin sekä varsinaiseen sektiossa kuljetettavaan hyötykuormaan.

`DVBStreamData`-luokka mallintaa MPEG-2-virrasta luettuja TS-paketteja. Lisäksi se sisältää virtaa koskevia tilatietoja.

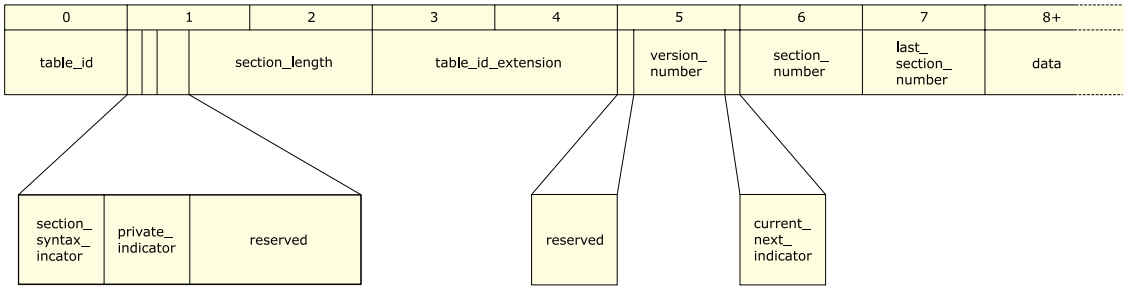
5.6 Sektioiden suodatus siirtobittivirrasta

Siirrettäessä dataa DVB-järjestelmässä, käytetään usein MPEG-2-sektioihin pohjautuvia protokollia. DVB:n määrittelemät data- ja objektikarusellit sekä palvelutiedon välitys ovat hyvä esimerkki tästä. MPEG-2-virrassa kulkevat sektiot identifioidaan 8-bittisellä taulutunnisteella (table_id). Palvelutietoon kuuluville tauluille on valittu standardoidut tunnisteet; esimerkiksi PAT:lle on varattu tunniste 0x00 ja PMT:lle 0x02. [8], [11]

Sovellusohjelma on yleensä kiinnostunut vain tietyn tyyppisistä sektioista. Jotta sovelluksen ei itse tarvitsisi lukea ja jäsentää siirtobittivirrassa kulkevaa dataa, on kehitetty mekanismeja, joilla sovellus voi määritellä, minkä tyyppisistä sektioista se on kiinnostunut. Näiden mekanismien avulla sovellus voi antaa alemman kerroksen tehtäväksi suodattaa halutun kriteerin täyttävät sektiot siirtobittivirrasta.

Sektioiden suodattaminen tapahtuu vertaamalla sektion otsikossa olevia tavuja suodattimen tavuhin. Vaikka eri tyyppisten sektioiden otsikot ovat sisällöltään eri-

[1] `org.davic.mpeg.sections.Section` [24].



Kuva 5.3. MPEG-2-sektion otsikon vakiokentät [24].

laisia, jokaisen sektion otsikon alussa on vakiokentät sisältävä kahdeksan tavun mittainen osa. Kuvassa 5.3 on esitetty sektion otsikon vakiomuotoinen alku.

Multimedia Home Platformissa käytettävä sektioiden suodatusmekanismi perustuu DAVIC-organisaation määrittelemään menetelmään. Menetelmässä suodatin koostuu taulutunnisteesta ja kahdesta 64 bitin (8 tavua) mittaisesta sekvenssistä. Ensimmäistä sekvenssiä nimitetään arvoksi (value) ja toista maskiksi (mask). Maski kertoo mitä bittejä vertailuun käytetään ja arvo kertoo mitkä arvot vertailuun käytettävillä biteillä tulee olla. Suodattimen tavuja verrataan sektion tavuihin 3–10. Sektion tavussa 0 on taulutunniste eivätkä tavuissa 1 ja 2 olevat kentät sisällä mitään sovellusten kannalta oleellista tietoa. Tavujen 8–10 käyttö suodatuksessa antaa mahdollisuuden sektioiden suodattamiseen datan sisällön mukaan. [24]

Käytettäessä positiivista suodatinta se päästää läpi sektiot, jotka toteuttavat ehdon[2]:

$$value \,\&\, mask = section\_header \,\&\, mask$$

Negatiivista suodatinta voidaan käyttää, kun halutaan suodattaa sektioita, joissa on tapahtunut jokin muutos, esimerkiksi version_number -kenttä on muuttunut.

[2] & tarkoittaa bittikohtaista AND-operaatiota.

Negatiivinen suodatin päästää läpi sektiot, jotka toteuttavat ehdon:

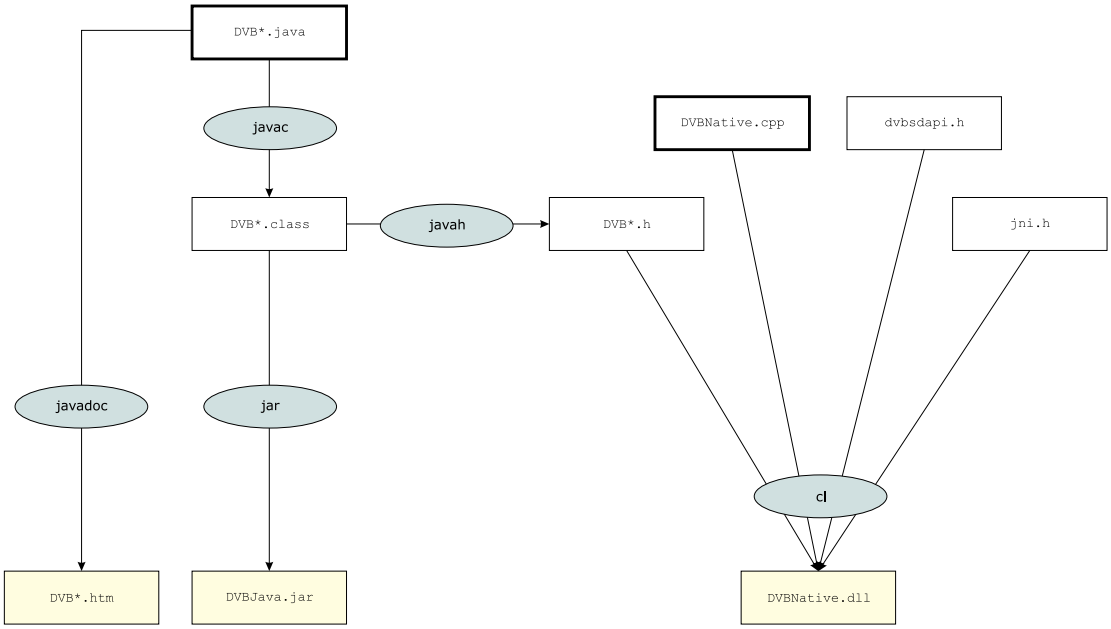
$$value \,\&\, mask \neq section\_header \,\&\, mask$$

Suodatusehtona voidaan käyttää myös positiivisen ja negatiivisen suodattimen yhdistelmää.

5.7 Toteutus

Ohjelmisto koostuu 26 Java-tiedostosta sekä yhdestä C++-tiedostosta, joka sisältää kaikki ohjelmiston toteutuksessa tarvittavan natiivikoodin. Kuvassa 5.4 on esitetty ohjelmiston kääntäminen lähdetiedostoista kohdetiedostoiksi. Java-lähdekooditiedostot (`DVB*.java`) käännetään Java-kääntäjällä (javac) tavukooditiedostoiksi (`DVB*.class`) jotka kirjastoidaan jar-työkalulla yhdeksi kirjastotiedostoksi (`DVBJava.jar`). Natiivikoodin toteutuksessa tarvittavat funktioesittelyt (`DVB*.h`) generoidaan javah-työkalulla tavukooditiedostoista. Natiivikoodin sisältävä C++-lähdekooditiedosto käännetään C++-kääntäjällä (cl) dynaamisesti ladattavaksi kirjastoksi (`DVBNative.dll`). Käännöksessä tarvitaan vastaanotinkortin rajapinnan määrittelyt (`dvbsdapi.h`) sekä JNI:hin liittyvät määrittelyt (`jni.h`). Java-rajapinnan HTML-muotoinen dokumentointi generoidaan javadoc-työkalulla Java-lähdekooditiedostoista. Koko ohjelmiston käännökseen käytetään make-työkalua, jonka avulla on helppo hallita tiedostojen oikea käännösjärjestys.

Java-osuuden kehittämiseen käytettiin Sun Microsystemsin Java Development Kitiä ja natiivikoodi kehitettiin Microsoft Visual C++-kääntäjällä. Ohjelmisto on testattu JDK:n versioilla 1.1.8 ja 1.2.2. C++-koodi on testattu Microsoftin kääntäjän versiolla 5 ja 6. C++-koodi on C++-standardin mukaista, joten myös muiden kään-



Kuva 5.4. Ohjelmiston kohdetiedostojen tuottaminen lähdetiedostoista.

täjien käytön pitäisi onnistua. Tätä ei kuitenkaan ole testattu. Ohjelmiston pitäisi myös toimia muissakin Java-virtuaalikoneissa, jotka toteuttavat JNI-spesifikaation, mutta tätäkään ei ole käytännössä testattu.

5.8 Ohjelmiston käyttö

Rajapintaohjelmistoa käyttävät sovellukset saattavat olla hyvinkin erilaisia, mutta rajapinnan käytön kannalta ne sisältävät yleensä seuraavat toiminnot:

1. Alustustoimenpiteet
2. Tarvittavien Stream-objektien luonti ja niihin liittyvien parametrien asettaminen

3. Datan vastaanotto ja vastaanottoparametrien muutokset (esim. suodattimien parametrien muutokset)

4. Lopetustoimenpiteet, joihin sisältyy mm. avattujen virtojen sulkeminen ja varattujen resurssien vapauttaminen.

Alustustoimenpiteiden ensimmäisenä vaiheena on luoda järjestelmään asennettuja vastaanotinkortteja mallintavat objektit. Se tapahtuu `DVBController`-luokan `enumAdapters()`-metodin avulla.

```
DVBAdapter adapter = DVBController.enumAdapters()[0];
```

Tässä `adapter`-muuttujaan sijoitetaan ensimmäistä löytynyttä vastaanotinkorttia vastaava objekti. Ennen kortin käyttöä, on kutsuttava `open()`-metodia, joka alustaa kortin. Seuraavana toimenpiteenä on vastaanottoparametrien asettaminen. `setupReceiver()`-metodi asettaa antennia koskevat parametrit ja `setupChannel()`-metodi kanavakohtaiset parametrit mm. taajuuden, symbolinopeuden ja virheenkorjausparametrit.

```
adapter.setupReceiver(new DVBReceiverParameters());
adapter.setupChannel(new DVBChannelParameters());
```

Parametrien asetuksen jälkeen voidaan `getChannelInfo()`-metodilla hakea kanavaa koskevia tilatietoja. Tilatiedoista nähdään mm. onko vastaanotin lukkiutunut lähetyssignaaliin. Vastaanottoa varten luodaan `DVBRawStream`-objekti (tai jokin muu stream-tyyppinen objekti) ja se kytketään `adapter`-objektiin `attach()`-metodilla.

```
DVBRawStream stream = new DVBRawStream(400)
stream.attach(adapter);
```

Tämän jälkeen vastaanotto aktivoidaan `enable()`-metodilla ja dataa luetaan `getData()`-metodilla.

```
stream.enable();
DVBStreamData sd = null;
do {
  sd = stream.getData(1);
  if(sd.data != null)
    System.out.println("TS Header: " +
                       sd.data[0] + ":" + sd.data[1] + ":" +
                       sd.data[2] + ":" + sd.data[3]);
} while(readMore());
```

Ohjelman lopuksi suljetaan yhteys vastaanotinkorttiin ja vapautetaan varatut resurssit.

```
stream.disable();
stream.detach();
adapter.close();
```

Liitteessä A on esitetty kokonaisuudessaan esimerkkiohjelma, joka suodattaa PAT-sektioita MPEG-2-virrasta ja tulostaa niiden pituuden.

Luku 6

Yhteenveto

Digitaalinen tekniikka on vallannut monet tietotekniikkaan liittyvät ja sitä hyödyntävät alat ja se on mahdollistanut esimerkiksi maailmanlaajuiset tietokone- ja puhelinverkot. Televisioon liittyvä jakeluverkko on kuitenkin saanut odottaa digitalisointia pitkään, vaikka TV-studioissa tapahtuva tiedon tallennus ja siirto onkin jo suurimmaksi osaksi siirtynyt digitaaliaikaan. Syynä tähän on se, että digitaaliseksi bittivirraksi muutetun TV-kuvan käsittely kotipäätteissä on tullut mahdolliseksi vasta viime vuosina.

Digitaalinen lähetystekniikka mahdollistaa osittain Internet-palveluiden sekä monien muiden vuorovaikutteisten lisäpalvelujen välittämisen televisioverkossa. Internetin huima kasvuvauhti ja sen suuret käyttäjämäärät osoittavat, että kuluttajilla on kiinnostusta uudenlaisten vuorovaikutteisten palvelujen käyttöön. Kotitietokoneiden korkea hinta ja hankala käytettävyys ovat kuitenkin monelle esteenä Internetin kotikäyttöön. Digitaalista televisiota onkin markkinoitu monissa yhteyksissä laitteeksi, joka tuo Internetin ja vuorovaikutteiset palvelut jokaiseen kotiin.

Suomessa käyttöönotettavat digitaaliset TV-vastaanottimet perustuvat DVB-organisaation määrittelemään Multimedia Home Platform -määrittelyyn. MHP:n mu-

kaisissa vastaanottimissa vuorovaikutteiset sovellukset toteutetaan Java-sovelluksina, jotka välitetään katsojille TV-lähetteen mukana tai paluukanavaa pitkin.

Java-sovellusten välittämiseen TV-lähetteen mukana käytetään joko data- tai objektikaruselliprotokollia. Koska MHP:n mukaisia digitaalisia vastaanottimia ei ole vielä saatavilla, data- ja objektikarusellien käyttöön liittyvää tutkimustyötä on tehty Digitaalisen median instituutissa kehitetyllä PC-pohjaisella testialustalla. Testialustan toteutus perustuu tässä työssä kehitettyyn TV-vastaanotinkortin Java-rajapinnan käyttöön.

Tutkimustyössä on perehdytty karusellien yleiseen toimintaan ja ominaisuuksiin. Testialustan avulla on myös tutkittu vuorovaikutteisten palvelujen, jotka on toteutettu Java-sovelluksina, välittämistä TV-lähetteen mukana. Samalla on selvitetty palvelujen ja niihin liittyvän datan kaistavaatimuksia.

Tässä työssä toteutettu TV-vastaanotinkortin Java-rajapinta on toteuttanut sille asetetut vaatimukset. Testialustaan toteutetut data- ja objektikarusellien vastaanottoon käytettävät ohjelmistot voitiin rakentaa vaivattomasti vastaanotinkortin rajapintaohjelmiston päälle. Vaikka rajapintaohjelmistoa käytetäänkin pääasiassa melko hitaiden bittivirtojen vastaanottoon, ei testeissä havaittu suorituskykyongelmia esimerkiksi ääntä ja kuvaa sisältävien nopeiden bittivirtojen vastaanotossa.

Testialustaa on tulevaisuudessa tarkoitus laajentaa vastaamaan enemmän todellista vastaanotinta toteuttamalla tarvittavat MHP:n rajapinnat. Mukaan liitetään mm. toimikortin (esimerkiksi Suomessa käytettävä HST-kortti) ohjaamiseen liittyvät rajapinnat, kuvan ja äänen esittämiseen tarvittavat komponentit (Java Media Framework) ja Xlet-sovellusten suorittamiseen tarvittava ajoaikainen ympäristö.

Lähdeluettelo

[1] Ari Ikonen. *Digitaaliohjelmaa TV-satelliiteista.* Omakustanne, 1998.

[2] Suomen Digi-TV-Forum. *Suositus suomalaisen digitaalitelevision vastaanottimesta (lehdistötiedote)*, Huhtikuu 1999.
⟨URL:http://www.digitvforum.fi/ ⟩.

[3] Digi-TV-työryhmä. *Digitaalinen televisio ja Suomi.* Liikenneministeriön julkaisuja 23/98, Toukokuu 1998.

[4] Winston William Hodge. *Interactive television: a comprehensive guide for multimedia technologists.* McGraw-Hill, 1994.

[5] Kari Jääskeläinen. *Interaktiivisen television sisällöntuotanto.* Espoo Enterprises Oy, 1997.

[6] NorDig. *NorDig I: Digital Integrated Receiver Decoder Specification.* NorDig, 1.2 edition, 1999.
⟨URL:http://www.nordig.org/specifications.html ⟩.

[7] Kaj Södergård, Ville Ollikainen, and Risto Mäkipää. *Digitaalisten televisiolähetysten käyttö datajakelussa.* Valtion teknillinen tutkimuskeskus, Maaliskuu 1999.

[8] ISO/IEC 13818-1. *Information technology — Generic coding of moving pictures and associated audio information: Systems.* International Standard. ISO/IEC, 1st edition, 1996.

[9] Barry G. Haskell, Atul Puri, and Arun N. Netravali. *Digital Video: An Introduction to MPEG-2.* Digital Multimedia Standards Series. Chapman & Hall, September 1996.

[10] Hervé Benoit. *Digital Television: MPEG-1, MPEG-2 and Principles of the DVB System.* John Wiley & Sons, Inc., New York, 1998.

[11] European Telecommnications Standards Institute. *Digital Video Broadcasting (DVB); Specification for Service Information (SI) in DVB systems.* ETSI, V1.3.1 edition, 1998.

[12] European Telecommnications Standards Institute. *Digital Video Broadcasting (DVB); DVB specification for data broadcasting.* ETSI, V1.1.1 edition, 1997.

[13] Digital Video Broadcasting. *SI-DAT 382: Implementation Guidelines for Databroadcasting.* Digital Video Broadcasting, 8th edition, 1998.

[14] ISO/IEC 13818-6. *Information technology — Generic coding of moving pictures and associated audio information — Part 6: Extensions for DSM-CC.* International Standard. ISO/IEC, 1st edition, 1998.

[15] Object Management Group. *Common Object Request Broker: Architecture and Specification.* Object Management Group, 2.0 edition, 1995.

[16] Digital Video Broadcasting. *Multimedia Home Platform*, 1999. Revision 11.

[17] Jean-Pierre Evain. *The Multimedia Home Platform — an overview.* EBU Technical Review, No. 275, 1998.

[18] Georg Luetteke. *The DVB Multimedia Home Platform*, 1999. ⟨URL:http://www.dvb.org/ ⟩.

[19] Allen Mornington-West. *MHEG-5 and Java — the basis for a common European API?* EBU Technical Review, No. 275, 1998.

[20] Gary Cornell and Cay S. Horstmann. *Core Java.* Java Series. The Sunsoft Press, 2nd edition, 1997.

[21] Kai Koskimies. *Pieni oliokirja.* Suomen Atk-kustannus Oy, Espoo, 1997.

[22] Tim Lindholm and Frank Yellin. *The Java Virtual Machine Specification.* Addison-Wesley, 1999.

[23] Patric Chan, Rosanna Lee, and Douglas Kramer. *The Java Class Libraries.* Addison-Wesley, 2nd edition, 1997.

[24] Digital Audio-Visual Council. *DAVIC 1.4 Specification Part 9: Information Representation (Technical Specification).* Digital Audio-Visual Council, 1998. ⟨URL:http://www.davic.org/ ⟩.

[25] HAVi Group. *The HAVi Specification.* HAVi Group, 1.0 beta edition, 1998. ⟨URL:http://www.havi.org/ ⟩.

[26] Sun Microsystems. *Java Media Framework Specification*, November 1999. ⟨URL:http://java.sun.com/products/java-media/jmf/ ⟩.

[27] Sun Microsystems. *Java Native Interface Specification*, May 1997. ⟨URL:http://java.sun.com/products/jdk/1.2/docs/guide/jni/ ⟩.

Liite A

Esimerkki Java-rajapinnan käytöstä

DVBTest.java

```
import dmi.cc1016.*;
import java.io.*;

/**
 * This example program demonstrates the reception of MPEG-2 sections
 * with a COCOM CC1016 receiver card and the Java API
 *
 * @author Olli Savia
 *
 */
public class DVBTest {
  private static DVBAdapter adapter = null;

  public static void main(String args[]) {
    try {
      // Find all adapters installed in the system and
      // use the first one available
      adapter = DVBController.enumAdapters()[0];

      // Print Java API version information
      System.out.println(DVBController.getJavaInterfaceInfo());

      // Open adapter
      adapter.open();

      // Configure the receiver using the default parameters
      adapter.setupReceiver(new DVBReceiverParameters());

      // Set up a channel using the default parameters
      adapter.setupChannel(new DVBChannelParameters());
```

```
32      // Wait until the receiver is locked to the transmission signal
        DVBChannelInfo ci = null;
34      do {
          ci = adapter.getChannelInfo();
36      } while(ci.isLocked() == false);

38      // Create an MPEG-2 section stream with 400 packets buffering capacity
        DVBSectionStream stream = new DVBSectionStream(400);
40
        // Attach the stream to the adapter
42      stream.attach(adapter);

44      // Set a filter to receive program association table sections
        // from the stream. PAT's PID is always 0x0000 and table_id is 0x00
46      stream.addPID(0x0000);
        stream.setFilter(0x00);
48
        // Start receiving
50      stream.enable();

52      DVBSection section = null;

54      // Get 10 sections from the stream and display
        // the value of section_length field of each section
56      for(int i = 0; i < 10; ) {
          try {
58          section = stream.getSection();
            System.out.println("section_length: " + section.section_length());
60          ++i;
          } catch(DVBNoDataAvailableException x) {}
62      }

64      // Release all acquired resources
        stream.disable();
66      stream.detach();
        adapter.close();
68    }
      catch(Exception e) {
70      System.out.println(e);
      }
72  }
}
```