Tidsförflyttning och valfri åtkomst i strömmande MPEG-video över HTTP

Jan-Erik Skata

Diplomarbete i inbyggda system
Handledare: Jerker Björkqvist
Laboratoriet för inbyggda system
Tekniska fakulteten
Åbo Akademi
Datum: den 29 maj 2007

Sammanfattning

Målet med detta diplomarbete har varit att undersöka möjligheterna att störningsfritt förflytta positionen för uppspelning i strömmande MPEG-video samt att förflytta uppspelningspositionen önskat lång tid över HTTP. Implementationen skall användas i ett Personal Video Recorder-system för IPTV.

Sökord: IPTV, MPEG, HTTP, strömmande video, VOD, video on demand, PVR, personal video recorder, digital-TV.

Innehåll

Inledning

Valet av diplomarbete var för mig beroende av ett antal faktorer; kanske viktigast var att göra någonting som skulle komma till praktisk nytta – att lösa ett realistiskt tekniskt problem. Jag var inte intresserad att utveckla någonting som endast skulle fungera som demonstration av ett teoretiskt koncept. Att utveckla en produkt som förhoppningsvis faktiska hemanvändare dagligen kommer att använda är för mig en sak som gör arbetet mycket mera intressant.

En annan sak som givetvis var viktigt var anknytningen till huvudämnet för mina studier, inbyggda system. Förenklat kan man säga att ämnet innebär utveckling av programvara för ett specialiserat system, ofta mycket närmare relaterat till hårdvaran än vid generell mjukvaruproduktion.

I synnerhet digital-TV och interaktiv TV tycker jag personligen är väldigt intressant. Jag hade också möjlighet att göra diplomarbetet på forskningsinstitutet Devera där jag var anställd, och vars verksamhet till stor del är fokuserad på digital-TV, interaktiv TV samt teknologier för överföring av bild och ljud. I början av oktober 2006 kom jag i kontakt med företaget Hibox Systems som drivs av två av mina bekanta, Staffan Granholm och Andreas Öhman, och fick reda på att de hade idéer som kunde förverkligas som diplomarbete.

Hibox Systems, grundat 2005, är ett företag i Åbo som specialiserar sig främst på IPTV-lösningar, dels för väl avgränsade miljöer som exempelvis hotell, dels för hemanvändning och kabelnät. Man har än så länge designat IPTV-system för den finländska hotellkedjan Omena Hotellit samt för kabel-TV-operatören Närpes Centralantennandelslag i samarbete med Internetleverantören Dynamo Net.

Hibox Systems presenterade ett antal intressanta projekt, och vi bestämde oss slutligen för att undersöka och om möjligt lösa problemet med snabbspolning i HTTP-strömmande video. Detta projekt hade direkt anknytning till många av de kurser jag gått under ämnesstudierna, bland annat begrepp som MPEG-video, olika former av digital-TV, realtidsapplikationer och datakommunikation. Ämnet var också intressant från Deveras sida.

Projektet som presenterades var följande: Hibox Systems har implementerat en funktion i sin serverprogramvara för att spela in önskade TV-program. Detta är således en nätbaserad ersättare för videobandspelare; användaren ställer in vilka program som skall

spelas in med hjälp en meny i set-top-boxens användargränssnitt. Servern håller reda på vilka program som önskas spelas in och sparar ner dem när de sänds. Ingenting lagras på själva set-top-boxen, utan endast servern lagrar inspelningarna som filer.

När sedan användaren önskar se på de inspelade TV-sändningarna finns de tillgängliga i set-top-boxens användargränssnitt. Helt enligt IPTV-konceptets filosofi öppnar set-top-boxen en dataström från servern och spelar upp videomaterialet.

Detta är var det egentliga problemet börjar – man kan idag inte snabbspola framåt eller bakåt när videomaterialet spelas upp. Det understöds inte av de nätverksprotokoll som använts i implementationen av dataöverföringen. Den enkla exempelimplementation som gjorts gör att systemet blir förvillat, och är inte heller tänkt att användas i praktiken. En sådan funktion anses av vanliga användare vara självklar, och i synnerhet eftersom användaren inte känner till hur ett sådant system fungerar kan avsaknaden av funktionen leda till en del frustration. Förflyttningen skall helst också kunna göras med önskat antal sekunder.

Funktionen som eftersöks finns tillgänglig med hjälp av proprietära programpaket, men eftersom marknaden för IPTV-system är begränsad till några få väldigt stora aktörer är prisnivån högre än Hibox Systems är intresserad att betala.

Målet för mitt diplomarbete är därmed att utreda vilka möjligheter som finns för att implementera en funktionalitet för snabbspolning i strömmande video med hjälv av standardiserade protokoll och fri mjukvara, och om möjligt, förverkliga detta i verkligt driftssystem.

Kapitel 1

Television – teknisk bakgrund

1.1 Digital television

1.1.1 Från den analoga televisionen

Under andra halvan av 1980-talet insåg man att televisionssystemet höll på att bli hopplöst föråldrat. Mats Röjne [1] tar upp televisionsteknikens historia i “Digital-tv via mark, satellit och kabel”. Metoderna för analoga televisionssändningar har utvecklats utifrån 1930-talets teknologi. Den ursprungliga svartvita televisionen kompletterades med färgsändningar på 1950–1960-talen och stereoljud i slutet av 1980-talet, medan systemet fortfarande är fullständigt kompatibelt med alla gamla svartvita mottagare. Den första formen av interaktivitet, Teletext eller text-TV, introducerades på 1970-talet och har sedan dess funnits kvar i stort sett oförändrad.

En anledning till att man behövde utveckla nya tekniker för TV-sändningar var att de analoga marksändningarna upptar mycket stor bandbredd i frekvensspektret. Enligt Röjne [2] upptar en analog TV-sändning 7 eller 8 MHz, på VHF- respektive UHF-bandet. Eftersom frekvensutrymmet är begränsat ryms inte mera än ca 60 TV-kanaler (59 enligt den svenska frekvensfördelningen). Även om det är många flera TV-kanaler än vad som sänds för tillfället i Norden, har den tekniska utvecklingen gjort att man behöver radiokommunikation till allt flera användningsområden. Det kommer att bli nödvändigt att frigöra frekvenser för annan kommunikation. Detta, i kombination med höjda krav från konsumenterna på högre bildkvalitet utan störningar, har legat till grund för utvecklingen av digitala TV-system.

De analoga sändningarna håller på att stängas av världen över. I Finland kommer alla analoga sändningar att tystna 31.8.2007 (Digita [3]), vilket innebär att alla därefter måste använda någon form av digital mottagare för att ta emot TV-sändningar.

1.1.2 Mot digitala TV-sändningar

Som följd av insikten i det analoga systemets begränsningar började man i slutet av 1980-talet undersöka hur det kunde vara möjligt att sända television digitalt, med effektivare utnyttjande av frekvensspektret och bibehållen, eller helst högre, bildkvalitet.

Konceptet HDTV (High Definition TeleVision), TV-sändningar med signifikant högre upplösning än standard, blir samtidigt aktuellt. Än idag hålls HDTV tillbaka av den traditionella TV-tekniken, eftersom endast de nyaste TV-apparaternas skärmar klarar HDTV:s högre upplösning.

Man skall även komma ihåg att det är först mot slutet av 1990-talet som tekniken för digitala mottagare, s.k. set-top-box, har kommit ner i prisklasser som gjort dem intressanta för konsumenter, och tekniken kommit upp i den prestanda som krävs för att i realtid behandla strömmen av data.

1.1.2.1 I Europa

Ulrich Reimers [4] tar upp historien bakom systemen i “Digital Video Broadcasting (DVB); the international standard for digital television". Det europeiska DVB-projektet som officiellt startade i september 1993 är vad som har utvecklats till dagens digitala TV-teknik. Grunden till DVB-projektet lades i egentligen redan i slutet av 1991 och var ursprungligen en lös sammanslutning av företag, organisationer och myndigheter med intresse för digital television, med en majoritet tyska parter. DVB-projektet har idag över 260 medlemmar, inte bara från Europa utan från hela världen. En av DVB-projektets principer är öppna standarder, och tillverkare som uppfyller dessa kan enligt DVB-projektets officiella hemsida [5] anhålla om att få använda DVB-projektets symbol.

Reimers [4] behandlar att man i ett tidigt skede hade bestämt sig för att sammarbeta med MPEG (Moving Pictures Expert Group), vars system MPEG-2 för videokomprimering och MPEG Audio för ljud ansågs lämpliga för digitala TV-system. I början koncentrerade man sig på markbundna sändningar, vilket sammanfattades i rapporten “Prospects for Digital Terrestrial Television”, publicerad i november 1992 – detta ledde till det som idag heter DVB-T. Specifikationer för digitala sändningar över satellit (grunden till DVB-S) samt kabelnät (DVB-C) följde i november 1993 respektive januari 1994. Alla dessa utvecklades för att vara redo för såväl SDTV (Standard Definition TeleVision), standardformat, som HDTV.

1.1.2.2 I övriga världen

I USA inriktade man sig mera på HDTV än en digital standard för normalupplösta TV-sändningar. Redan 1987 utlyste amerikanska telekommunikationsmyndigheten FCC (Federal Communication Commission) en förfrågan om ett system för marksända HDTV-sändningar.

Man mottog 21 förslag, varav de flesta egentligen inte var användbara för högupplösta TV-sändningar. Kandidaterna kunde snabbt reduceras, och 1994 återstod fyra fullständigt digitala system. Reimers [4] berättar vidare att de fyra återstående inledde ett samarbete kallat “Great Alliance HDTV System”. Denna allians består av ATT/Zenith Electronics, General Instrument, Massachusets Institute of Technology och Philips/Thomson/David Sarnoff Research Center. Systemet som presenterades för marksändningar har antagits av ATSC (Advanced Television Systems Committee) och påminner mycket om DVB; man använder även här MPEG-2-video, dock med AC-3 som protokoll för audio.

I Japan hade man sedan tidigare en analog standard för HDTV, MUSE (MUltiple Subsampling Encoding), och man började, enligt Reimers [4], därför undersöka digitala lösningar först 1994. Det japanska systemet ISDB-T/S/C (Integrate Services Digital Broadcasting) som används idag är mycket likt de europeiska DVB-standarderna.

Den kanadensiska organisationen ABSOC (Advanced Broadcasting Systems Of Canada) har främst fungerat som en länk mellan ATSC/Great Alliance och det europeiska DVB-projektet. Man använder idag ATSC-systemet i Kanada [4].

1.1.2.3 Förberedelser för interaktivitet

Den första typen av interaktivitet, text-TV, började se föråldrad och begränsad ut när de digitala systemen utvecklades. Text-TV, eller Teletext, är en enkel envägskommunikation där textbaserade sidor sänds ut kontinuerligt vid sidan om bild- och ljudsändningen. Inga grafiska funktioner utöver olika färger på texten är möjliga och inte heller kommunikation tillbaka till operatören. Typiska användningsområden för Teletext är nyheter, väder, börskurser och programtablåer. Trots dessa begränsningar har systemet blivit mycket allmänt och de flesta TV-kanaler har Teletext-sändningar, samtidigt som i stort sett alla TV-mottagare kan ta emot dem.

I samband med utvecklingen av DVB-standarderna skulle även funktioner för bättre interaktivitet realiseras. Steven Morris och Anthony Smith-Chaigneau tar upp historien bakom MHP (Multimedia Home Platform) och OCAP (OpenCable Applications Platform) i “Interactive TV Standards” [6]. EU-projektet DG III – UNITEL var det ursprungliga initiativet till en interaktiv standard anpassad för digital-TV. Projektets förslag antogs i oktober 1997 av DVB-samarbetet. Steven Morris och Anthony Smith-Chaigneau beskriver MHP:s grundläggande idé som en plattformsoberoende miljö för interaktivitet och multimedia. Vidare kan nämnas det amerikanska OCAP som utvecklades endast för digital-TV över kabelnät. OCAP som ursprungligen utvecklades av det amerikanska företaget CableLabs, har idag mycket gemensamt, och är till stora delar kompatiblet, med MHP.

Morris och Smith-Chaigneau [6] behandlar vidare JavaTV, ett programmeringsgränssnitt från Sun Microsystems som utvecklats att vara lämpligt för interaktivitet och multi-

media. JavaTV antogs av DVB-gruppen som underliggande arkitektur i MHP 1.0. Denna arkitektur benämns som middleware – mellanliggande programvara, och beskrivs ingående av Morris och Smith-Chaigneau i kapitlet “Middleware Architecture” [7]. Även om arkitekturen innehåller en virtuell Java-maskin, är den bredare än så. Ett MHP-program kallas Xlet, och använder AWT (Abstract Window Toolkit) tillsammans med HAVi (Home Audio/Video Interoperability) för grafik- och audiofunktioner. En Xlet kan använda DVB-mottagarens returkanal (ofta telefonmodem eller Ethernet) för att ansluta över IP till en Internetansluten dator. Tack vare basen i Java är det lätt som programmerare att komma igång med MHP om man tidigare har erfarenhet av Java på persondatorer. I likhet med text-TV och analoga TV-system sänds MHP och OCAP ut tillsammans med den specifika kanal de är associerade till.

Enligt Morris och Smith-Chaigneau [6] var Finland först med att börja sända ut MHP-tjänster, detta var 2002. Hårdast har kanske Italien satsat på MHP, med hjälp av statligt subventionerade mottagare. MHP-organisationens statusrapport från september 2006 [8] visar att Italien fortfarande är mest aktiva med ca 3,9 miljoner mottagare. I Finland är intresset än så länge svalt, med ca 50 000 MHP-kompatibla mottagare. Många tjänster som utnyttjar MHP påminner om text-TV – exempelvis nyheter och väder. Utöver detta finns ett antal nya koncept som inte hade varit möjliga med text-TV – MHP-rapporten [8] nämner publikröstningar, telefonkataloger, chatsystem och biljettbokningssystem.

1.1.3 Samtidigt börjar media sändas på Internet

Redan i början av 1980-talet börjar entusiaster skaffa hemdatorer och koppla upp sig mot andra datorsystem. Uppringda modemförbindelser till s.k. BBS (Bulletin Board System) var bland dessa pionjärer ett populärt sätt att diskutera och utbyta filer. Konceptet med elektronisk post började ta form, och vissa BBS tillhandahöll utöver intern, en Internetansluten e-mail.

I mitten av 1990-talet började Internet etableras även bland vanliga hemanvändare. Hemdatorernas beräkningsprestanda och möjligheter till audio- och videomedia ökade, på samma gång som tekniken blev billigare. Även anslutningarna till Internet blev snabbare och billigare, i synnerhet från början av 2000-talet när DSL-anslutningarna (Digital Subscriber Line) började marknadsföras av teleoperatörerna. World Wide Web blev den dominerande informationskanalen och WWW blev för många användare, felaktigt, synonymt med Internet.

Utgående från dessa framsteg blev den naturliga fortsättningen att Internet och World Wide Web utvecklades från att ha innehållit mestadels statiskt innehåll såsom text och bilder till att innehålla dynamisk och direktsänd multimedia. I en artikel om nätradio på Wikipedia [9] berättas att 1993 gjorde Carl Malamud ett första försök med radio på

Internet. Till sitt försök använde han en form av IP Multicasting, MBONE. Den första egenliga nätradiostationen som artikeln på Wikipedia nämner, var Radio HK som drevs av reklambyrån Hajjar/Kaufman New Media Lab. Från att ha använt CU-SeeMee, ett proprietärt videokonferensssystem, övergick man senare till RealNetworks, även det ett proprietärt system för strömmande media. RealAudio-formatet blev snabbt dominerande för direktsända Internetradiostationer.

Från radio på Internet var steget till TV inte långt, speciellt som datakommunikationens bandbredd, datorernas prestanda samt användarnas intresse och förväntningar ökade. I en artikel på Wikipedia [10] nämns World News Now[1] från ABC som 1994 var de första nätsändningarna med video. Tekniken för TV på Internet varierar mycket, men även här är det proprietära RealPlayer stort. Ett antal kända TV-kanaler finns på Reals lista över betalkanaler[2], däribland EuroNews, MTV och BBC. Microsoft marknadsför i USA Internet-TV-tjänsten MSN TV[3], i form av en set-top-box som tekniskt sett påminner mycket om de boxar som används för IPTV. MSN TV kräver ett abonnemang från Microsoft och använder en befintlig bredbandsanslutning.

Utöver dessa finns ett antal tjänster som definitionsmässigt, tekniskt och juridiskt sett är "gråzoner", varav gratistjänsten YouTube[4] är mest känd. YouTube tillhandahåller möjlighet att lägga upp egna videoklipp för beskådan, men även här finns även en stor andel upphovsrättskyddat material olovligt upplagt. YouTube använder ett gränssnitt i webläsaren för att spela upp videomaterialet. Det kinesiska TVU networks[5] kan enligt Wikipedia [11] kallas P2PTV (peer-to-peer TV) och bygger på fildelningssystemet BitTorrent. TVU networks sänder ett stort antal kommersiella TV-kanaler, i många fall utan tillåtelse. Man använder en proprietär mediaspelare, TVUplayer, som tros vara baserad på den fria VLC Player[6] – detta i strid mot GPL-licensen. Ett liknande koncept är The Venice Project, senare benämnt Joost[7], som behandlas av Steve Rosenbush [12] i en artikel på BusinessWeek. Venice Project är den senaste idén från Niklas Zennström och Janus Friis, skaparna av fildelningssystemet Kazaa och Internet-telefonsystemet Skype. Enligt Rosenbush strävar Venice Project till ett nära samarbete med de TV-kanaler som skall sändas, i likhet med Apple Computers avgiftsbelagda musik- och filmtjänst iTunes[8]. Venice Project kommer att baseras på en peer-to-peer-baserad teknologi.

[1] http://abcnews.go.com/WNN
[2] http://www.real.com
[3] http://www.msntv.com
[4] http://www.youtube.com
[5] http://tvunetworks.com
[6] http://www.videolan.org
[7] http://www.joost.com
[8] http://www.apple.com/itunes

1.2 IPTV – television över datanätverk

Utöver digitala TV-sändingar i form av DVB-C i kabelnät började telekommunikationsoperatörer och Internetleverantörer undersöka möjligheterna för TV-sändningar över bredbandsanslutningar – som en jämförelse, rena motsatsen till konceptet med Internet över kabel-TV med s.k. kabelmodem. Med teknologier för snabba Internetanslutningar som exempelvis optisk fiber till hemmet öppnar sig möjligheter för nya tjänster. I bostäder som inte redan har kabel-TV kan det bli avsevärt dyrare att ansluta sig till kabel-TV-nätet jämfört med en Internetanslutning. Utöver detta öppnar IPTV – Internet Protocol TeleVision – konkurrensmöjligheter eftersom kabel-TV-operatörerna tidigare haft närmast monopolställning regionalt i sina kabel-TV-nät.

Broadband Services Forum har publicerat "IPTV Explained" [13] som beskriver naturen av ett IPTV-system. Man använder det välkända paketformatet IP (Internet Protocol) för att paketera TV-sändningar, i allmänt tillsammans med ett transportprotokoll som exempelvis UDP (User Datagram Protocol), för överföringen. Fördelarna kan enligt rapporten sammanfattas som enkelhet att distribuera större utbud av videomaterial till större antal tittare, samtidigt som den utnyttjade bandbredden hålls på en acceptabel nivå. Genom att använda IP-nätverk fylls på samma gång behovet av en returkanal, som ofta är nödvändig för interaktivitet.

Denna tvåvägskommunikation öppnar möjligheter till ett antal intressanta tilläggstjänster. Broadband Services Forum visar några exempel i rapporten "The IPTV Bundle" [14]. Man kunde marknadsföra en tjänst som låter användaren prenumerera på musik – för en fast avgift får man lyssna på valfri musik ur operatörens utbud så mycket man vill. Detta ställer dock krav på samarbetsvilja från skivbolag för den ekonomiska modellen. Enkla spel, i synnerhet kanske med flera spelare, kan förverkligas på IPTV-system. Dessa kan finansieras antingen med avgifter från användarna, eller med reklamintäkter. Rapporten [14] nämner videokonferens, videoövervakning för hemmet samt kontaktannonser som möjliga koncept för IPTV. Alla tre av dessa är redan välkända och i användning på World Wide Web, men erbjuder lukrativa affärsmöjligheter. Slutligen kan nämnas s.k. hypercontent, dvs. möjlighet för tittaren att få mera information om ett TV-program. Det kan finnas tillgängligt olika kameravinklar och statistik om exempelvis ett idrottsevenemang. Sådan typ av interaktivitet som extrainformation om ett TV-program är vad man ofta åstadkommer med MHP över DVB-system.

Konceptet med IPTV är inte helt olikt strömmande media över Internet. Dock är det viktigt att göra skillnad på IPTV och Internet-TV. Broadband Services Forum har sammanfattat skillnaderna och likheterna i rapporten "IPTV Versus Internet TV" [15]. Ett IPTV-nät är ofta avgränsat till ett mindre nätverk istället för att ha sändningarna fullt tillgängliga på Internet. Nätverket är dock ofta samma fysiska nätverk som används till

Internet-trafik, och World Wide Web kan vara tillgängligt i IPTV-mottagarens webläsare. En skillnad som rapporten nämner är att IPTV är tänkt att vara mycket mera robust med garanterad leverans av materialet. Eftersom nätet till viss del är privat och avgränsat är det lättare att hantera och man kan vara säker på att ha t.ex. tillräcklig bandbredd utan fördröjningar. Man har möjlighet att ge sändningar högre prioritet för att förhindra störningar i uppspelningen – vad som brukar benämnas Quality of Service (QoS).

Broadband Services Forums rapport [15] betonar vikten av att IPTV skall kännas som TV – allting skall fungera felfritt, omedelbart och kontrolleras endast med fjärrkontrollen. Med Internet-TV är det omöjligt att garantera att materialet levereras i realtid, vilket innebär att man måste mellanlagra en mängd data på mottagaren för att motverka avbrott i uppspelningen. En sak som nämns i rapporten är också åtkomstkontroll – i ett privat nätverk kan man enklare hålla reda på vilka mottagare som har rätt att ta del av materialet. Tack vare strukturen i t.ex. ett Ethernet-nätverk med växlar är det svårare att avlyssna material som man inte har rättighet till. Med kabel-, mark- och satellitsänd TV, såväl som Internet-TV, är materialet ofta tillgängligt för alla och man måste ta till lösningar med t.ex. kryptering för att motverka oauktoriserad visning.

1.2.1 IP multicast

I ett TV-nätverk har man ett stort antal klienter som alla skall kunna ta emot samma sändningar. Med analog och digital (DVB, ATSC) kabel-TV är detta problem obefintligt, eftersom båda använder sig av radiosignaler modulerade runt en bärfrekvens. Detta medför att nätverket till naturen uppför sig så att en signal når alla mottagare. Man kan dock vara tvungen att använda förstärkare för att få bra mottagning på stora avstånd. Med analoga televisionssignaler går det å andra sidan inte att sända en signal endast till en mottagare.

När man använder IP för att överföra TV-signaler som datapaket används oftast någon form av växlat nätverk som t.ex. Ethernet på datalänknivån. En nätverkstopologi med växlar gör att ett paket traverserar ner genom nätets trädstruktur tills det når den klient som står addresserad som mottagare.

Ralph Wittmann och Martina Zitterbart [16] jämför multicast (sändning till många mottagare) och unicast (sändning till en mottagare) och pekar ut skillnaden, att med multicast behöver avsändaren bara sätta ut materialet en gång på nätverket, till skillnad från unicast där det måste sättas en gång per mottagare. Att använda unicast för att sända exakt samma material till många mottagare, som t.ex. i fallet med IPTV, skulle kräva enormt stor bandbredd. Därför är det nödvändigt att kunna nå en bestämd grupp mottagare med samma utsända data. Wittmann och Zitterbart [16] skriver, att denna replikering av paket är en uppgift för nätverkets infrastruktur. De mellanliggande systemen måste hålla reda på åt vilka system de skall vidarebefordra datapaket. Utöver unicast-metodens problem

med bandbredd och skalbarhet för stora grupper av mottagare, tar man upp frågan om fördröjningar – om man skall sända samma data väldigt många gånger kommer det att bli en tidsskillnad mellan när den första och den sista i listan av mottagare får materialet.

Juan-Mariano de Goyeneches HOWTO [17] från The Linux Documentation Project[9] beskriver tekniskt hur IP multicasting fungerar. Med IP multicasting skickar avsändaren materialet till endast en IP-address, och alla som anmält sig som mottagare får det levererat. IP multicast använder sig av IP-addresser av Klass D, dvs. 224.0.0.0–239.255.255.255. Denna addressering kanske verkar mera logisk om man ser på IP-addressens 32 bitar, där en multicast-address kommer att ha första fyra bitarna 1110. IGMP (Internet Group Management Protocol) används av klienter för att ansluta sig till multicast-grupper. IGMP öppnar en rutt för multicast-paket från sin närmast router hela vägen till källan. Alla dessa routrar måste understöda IGMP för att det skall fungera, och det gör inte alla routrar idag.

Med tanke på att IPTV-nät i allmänhet är ett avgränsat nätverk – routrar kan väljas kompatibla med IGMP-protokollet – och att samma material skall sändas ut till många mottagare lämpar sig IP multicasting väl som nätverksprotokoll.

1.2.2 Video On Demand

Video On Demand är ett koncept som från kundens synvinkel, och marknadsföringsmässigt sett, påminner mycket om videouthyrning; skillnaden är att användaren kan beställa önskad video direkt vid sin TV eller dator, utan att behöva uppsöka en videouthyrningsbutik och utan att behöva lämna tillbaka filmen.

Enligt en artikel om Video On Demand på Wikipedia [18] gjorde Hongkong Telecom det första försöket med Video On Demand i början av 1990-talet. Det slog inte igenom speciellt bra p.g.a. teknologiska hinder och bristande intresse. I London inleddes lokala VOD-tjänsten HomeChoice från företaget Video Networks Limited, 1999. I USA startade Oceanic Cable på Hawaii sin VOD-tjänst i januari 2000. Som nordisk pionjär kan nämnas SF Anytime[10] med sin PC-baserade VOD-tjänst som använder Windows Media Player. Vasa Läns Telefon[11] samt Multi.fi[12] kan nämnas som finländska nätoperatörer som samarbetar med SF Anytime.

Pyssysalo och Ojala [19] beskriver tre typer av VOD (Video On Demand); full, near och adaptive. Full Video On Demand betyder att varje klient omedelbart efter förfrågan tilldelas en videoström, och ofta har möjligheter att pausa och snabbspola videouppspelningen. Near Video On Demand innebär att samma innehåll spelas upp med jämna mel-

[9] http://www.tldp.org
[10] http://www.sf-anytime.com
[11] http://www.netikka.fi/index.php?sivu=sf-anytime
[12] http://vod.multi.fi

lanrum. Efter att ha gjort förfrågan om att se en film kan tittaren vara tvungen vänta en stund innan filmen börjar sändas. Slutligen, Adaptive Video On Demand, innebär att det material som har störst efterfrågan spelas upp. Önskemål om material som inte fått tillräckligt stor efterfrågan kommer att förnekas. I de båda senare metoderna har tittaren ingen möjlighet att påverka uppspelningen.

Video On Demand är ett centraliserat system med en server som klienterna ansluter till. VOD-servern i ett IPTV-system kan vara antingen ett fristående datorsystem eller ett lastbalanserande s.k. kluster av flera datorer. Från VOD-servern strömmas allmän statisk video, t.ex. långfilmer som faktureras per visning. VOD-servrar använder i allmänhet RTP som protokoll för överföringen med RTSP för att styra dataströmmen – exempelvis att snabbspola framåt och bakåt. VOD-servrar köps ofta som färdig lösning med hårdvara och mjukvara färdig att ansluta till nätverket. Några leverantörer av sådana system som kan nämnas är Bitband[13], Tandberg[14], Entone[15] samt Anevia[16].

1.3 IPTV-system idag

1.3.1 Serversida

Serversidan spelar en betydande roll i ett IPTV-system. Broadband Services Forum beskriver i rapporten “IPTV Explained” [13] serversidans uppgifter som att ta emot TV-sändningar eller annan media som önskas sändas på IP-nätet, eventuellt omvandla dem till annat videoformat och förmedla dem till klienterna. Detta kan i praktiken innebära att ta emot sändningar som DVB-transportström, dela upp dem i olika programströmmar och mata ut i nätverket paketerat som UDP över IP Multicast. Beroende på system sänds programströmmarna som rå UDP-data eller RTP över UDP.

Video On Demand med statisk media lagrad på disk är en annan uppgift för IPTV-servermiljön, såväl som att sköta om registrering och debitering för avgiftsbelagda tjänster. Utöver sin egna huvudsakliga serverprogramvara kan operatören även ha en separat Video On Demand-server, vilka finns att köpa som färdig kommersiell lösning. IPTV-systemets anslutningsorienterade natur gör att full Video On Demand är att föredra, även om det ställer krav på bandbredd till klienterna.

Det mesta som syns från klientens synvinkel kommer från servern. Hela användargränssnittet laddas ned, ofta i XML-format, med tillhörande bild-data för bakgrund, logotyper och symboler. Således finns här alla möjligheter för operatören att skapa ett enhetligt och kundnära gränssnitt. I användargränssnittet finns i allmänhet tillgängligt en EPG,

[13] http://www.bitband.com
[14] http://www.tandberg.net
[15] http://www.entone.com
[16] http://www.anevia.com

Electronic Program Guide, som innehåller programtablåer för de en skilda kanalerna. Där kan finnas nyheter, väder och speltjänster, samt PVR-funktioner (Personal Video Recorder) för att spela in önskade sändningar som finns listade i programtablån, och spela upp tidigare inspelade program. Det är fullt möjligt att göra en PVR-funktion som möjliggör att när som helst under pågående program starta inspelningen för att kunna se resten senare.

Figur 1.1 illustrerar principen för ett IPTV-system med mottagning av digital-TV-sändningar, VOD-server och PVR-system. Flera IPTV-nät kan dela på realtidssändningarna som kan komma från en gemensam mottagare över Internet. Abonnenterna kan vara direkt anslutna eller över Internet.

Denna s.k. middleware, beskrivs av Broadband Services Forum [13] som en sorts portal som knyter ihop hela systemet. Observera att denna middleware är fullständigt orelaterad till den som nämnts i samband med MHP och OCAP. Även om det existerar färdiga lösningar att köpa är behoven så olika att middleware-mjukvaran ofta utvecklas av IPTV-operatören själv. Avsaknaden av standarder för IPTV har gjort det svårt att göra en generell middleware, utan operatören är närmast tvungen att skräddarsy sin serverprogramvara enligt den övriga systemlayouten – alternativ köpa dessa tjänster av en middleware-utvecklare. Användarterminalerna som är klienter i denna klient-server-modell, är ofta slutna system som är svåra att ändra, och därför måste serversidan vara anpassad till dem.

På servern finns även lagrat operativsystemet och övrig programvara som används av klienterna. Användarterminalerna ansluter t.ex. vid uppstart till servern och laddar vid behov ned ny systemprogramvara.

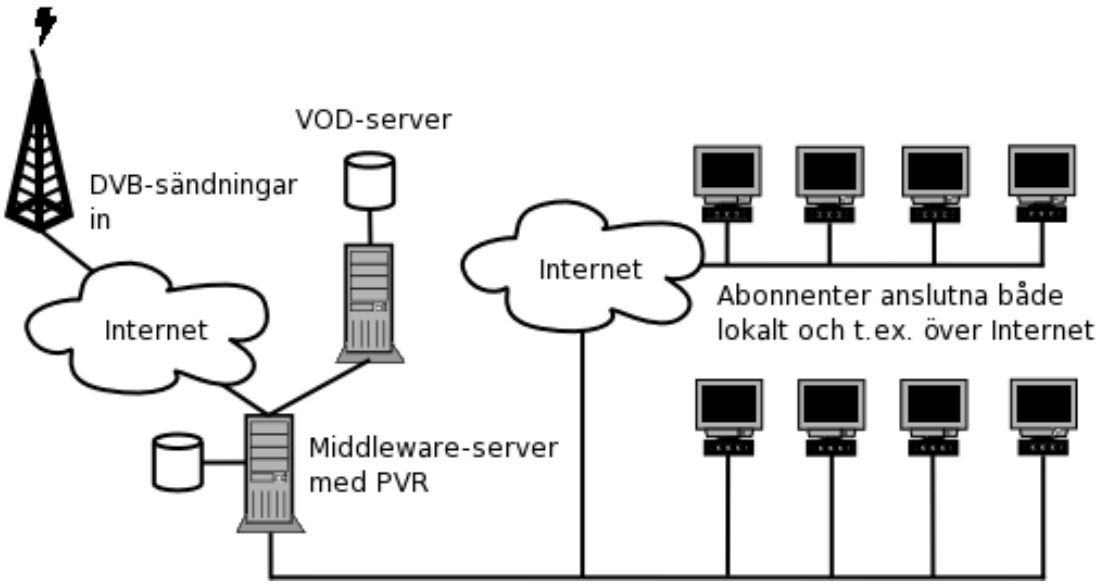
Ett försök till komplett helhetslösning som kan nämnas är Microsofts IPTV-system[17] som utvecklas i samarbete med Alcatel. Microsoft levererar all mjukvara för såväl server som klient samt middleware-gränssnitt. Systemet använder Windows Media som videoformat samt operativsystemet Windows CE för terminalerna.

1.3.2 Klientsida

Klienterna – användarterminalerna – består av specialiserade s.k. set-top-boxar som enbart är avsedda för att ta emot information i form av IP-data. Till skillnad från många andra set-top-boxar har en IPTV-mottagare i allmänhet ingen mottagare (tuner) för radiosignaler (avseende analoga såväl som digitala TV-signaler), utan endast ett Ethernet-gränssnitt. Användaren styr sin terminal med fjärrkontroll precis som många andra liknande system, men tangentbord finns även tillgängliga. I vissa fall används ett s.k. virtuellt tangentbord på skärmen för att mata in text.

Dessa mottagare är i grunden vanliga datorer med en processor dimensionerad att

[17] http://www.microsoft.com/tv/IPTVEdition.mspx



Figur 1.1: Typisk uppbyggnad för ett IPTV-system, med central middleware-server och VOD-system

endast vara tillräckligt kraftfull för sin uppgift, de innehåller en begränsad mängd arbetsminne samt ett icke-flyktigt minne som kan lagra en lokal kopia av systemprogramvaran. Även om terminalerna laddar sin systemprogramvara från servern vid uppstart kan uppstarten göras snabbare om man lagrar mjukvaran lokalt i ett icke-flyktigt flash-minne. Terminalernas mjukvara kan fjärruppdateras enligt önskemål från serverns administrationsgränssnitt, samt av terminalen själv genom att kontrollera om en nyare version finns tillgänglig.

Mottagarna har funktioner för att presentera ett användargränssnitt, vilket definieras från servern. Från användargränssnittet har användaren tillgång till bl.a. programtablåer, VOD-tjänster samt inspelning och uppspelning med den personliga inspelningsfunktionen. Ofta finns tillgång till World Wide Web med en enkel webläsare.

Leverantörer av IPTV-mottagare finns, precis som för VOD-system, idag en handfull. De mest betydande är amerikanska Motorola[18], Kreatel[19] – ett ursprungligen svenskt företag som var en av pionjärerna på IPTV. Enligt Wikipedia [20] uppköpt av Motorola 2006. Övriga som kan nämnas är Tilgin[20] – även det ett svenskt företag, norska Tandberg[21], brittiska Amino[22] samt franska Thomson[23].

[18] http://www.motorola.com
[19] http://www.kreatel.com
[20] http://www.tilgin.com
[21] http://www.tandberg.net
[22] http://www.aminocom.com
[23] http://www.thomson.net

1.3.3 Standardisering

Som tidigare nämnts, gör bristen på standarder för IPTV-system att alla operatörer blir tvungna att anpassa sin s.k. middleware enligt hur resten av systemet är uppbyggt. På samma sätt blir set-top-boxarna väldigt bundna till den IPTV-operatör som användaren abonnerar på. Om användaren då har köpt sin mottagare är sannolikheten liten att den går att använda i en annan operatörs IPTV-system, t.ex. om man flyttar eller av annan orsak väljer att byta operatör. Det enda som idag kan anses vara standard är användningen av IP-protokollet, RTP över UDP för realtidssändningar samt MPEG-formatet för video (som i allmänhet kommer direkt från en DVB-sändning).

1.3.3.1 Basboxprojektet

Som ett försök att sträva till en gemensam standard har det svenska forskningsinstitutet Acreo[24] utarbetat ett förslag på standardisering av set-top-boxarnas funktioner. Förslaget som sammanfattats av projektledaren Jari Aho [21] beskriver dels tekniskt en gemensam basnivå för mottagarnas funktioner och prestanda. Många av dessa punkter täcks redan av dagens mottagare – exempelvis DVB-standarden, MPEG, SDTV/HDTV-upplösningar samt att en standard-TV skall kunna användas för presentationen. Man försöker utöver detta sätta en gräns för hur länge det får ta att byta kanal – ner mot 0.5 s för MPEG-2. Några förslag på standardisering av EPG, text-TV och interaktivitet presenteras, bland annat MHP. För ljudet skall MPEG lager 1 och 2, MPEG-2 samt AC3 finnas eftersom de används i DVB. Utöver detta skall dubbelmono för olika språk samt vidarebefordran av Dolby Surround understödas. Stöd för Conditional Access-system via standardiserade gränssnitt samt Macrovision för att förhindra inspelning skall finnas.

Det kanske viktigaste med förlaget beskrivs i kapitel 1 – ett abstraktionslager som skall göra det möjligt för alla mottagare att kontakta samma IPTV-serversystem – dynamiskt och flexibelt är nyckelorden. Det skall även vara möjligt att via en gemensam portal få tillgång till funktioner från andra IPTV-leverantörer, exempelvis VOD-tjänster. På detta sätt skapas större valmöjligheter för kunden och större konkurrens bland operatörerna, som idag kunde anses ha något av en monopolställning regionalt. Detta abstraktionslager, SAL (Software Application Layer) som beskrivs närmare i kapitel 2 i Ahos förslag [21] är tänkt att ersätta behovet av tillverkarspecifika API:n för IPTV-mottagare.

I kapitel 4 diskuteras hur betalningar skall fungera. Dels är det nödvändigt att hålla reda på vilka mottagare som har rättighet att ansluta till nätverket överhuvudtaget, dels att fakturera för avgiftsbelagda tilläggstjänster från flera operatörer. Viktiga egenskaper för detta är, förutom säkerheten och integriteten, att det skall vara enkelt för användaren betala.

[24] http://www.acreo.se

1.4 Hibox Systems systemlayout

1.4.1 Serversida

Serversidan man använder idag är dels en egenutvecklad server, baserad på Java-servlets och körs på Apache Tomcat[25] och databassystemet MySQL[26]. Efter att set-top-boxens operativsystem startat upp, kontrolleras att den är registrerad och giltig i systemet, varefter användargränssnittet överförs. Det mesta av kommunikationen sker över HTTP, vilket innebär att man har väldokumenterade standarder att följa, samt ett bra underliggande system med TCP över IP för att garantera överföringen. Hibox Systems har ett fullständigt web-baserat administrationssystem där nätverksoperatören kan ställa in bland annat vilka terminaler (unika serienummer) som har rättigheter till nätverket, och vilka tjänster de kan använda.

I sin middleware har man implementerat funktioner bl.a. för EPG, som automatiskt uppdateras med listor tillgängliga från Internet. Man har en e-post-funktion som kan användas med ett externt e-post-konto, samt väderprognoser som hämtas från Internet. PVR-funktionen för att spela in önskade sändningar, kan ses som en nätansluten videobandspelare. Uppspelningen av dessa lagrade videoströmmar är vad som kommer att behandlas i detta diplomarbete.

Man använder en färdig VOD-lösning, en Vision 280 från BitBand. Denna är ett komplett system med hårdvara och mjukvara redo att kopplas in. VOD-servern är dock ett fullständigt slutet system – vad som kunde kallas en black box. BitBand tillhandahåller visserligen PVR-funktionalitet som tillägg men prisnivån gör att det är intressant att se på öppna lösningar. Hibox har för teständamål köpt denna utökade funktionalitet för begränsat antal användare.

1.4.2 Klientsida

IPTV-terminalen som skall användas som försöksplattform är IP-STB-serien från Kreatel. Terminalerna, som bygger på MIPS-arkitektur, kör ett Linuxbaserat[27] operativsystem. I systemet finns en X11-implementation[28] för de grafiska funktionerna, och webläsaren Mozilla[29] används för att besöka sidor på World Wide Web. Stora delar av systemet består således av fri mjukvara, även om programmet som spelar upp videoströmmen är proprietärt. Hibox Systems har införskaffat ett utvecklingspaket som möjliggör att skapa egna paket av systemprogramvara.

[25] http://tomcat.apache.org
[26] http://www.mysql.com
[27] http://www.linux.org
[28] http://www.x.org
[29] http://www.mozilla.org

1.4.3 Teknisk problemställning

Man har, som tidigare nämnts, implementerat en funktion i sin serverprogramvara för möjliggöra för användaren att spela in önskade TV-sändningar. Användaren kan med några enkla knapptryckningar på fjärrkontrollen välja program ur tablån (EPG) som skall lagras för framtida uppspelning. De lagrade programmen finns tillgängliga i användargränssnittet för att spelas upp när användaren önskar. Denna PVR-funktion fungerar alltså som en nätbaserad ersättare för videobandspelare.

Inspelningen fungerar – detta handlar endast om att skriva ned programströmmen till disk. Uppspelningen, som använder HTTP för att överföra videoströmmen fungerar också – bortsett från funktionen för att snabbspola framåt och bakåt i videoströmmen. Man har gjort ett försök med en enkel testimplementation på servern som endast hoppar framåt ett antal antal tecken i videofilen. Problemet är att p.g.a. MPEG-formatets struktur med I-, B- och P-rutor samt grupper av bilder (GOP) gör denna metod att mottagarens MPEG-avkodare blir “förvillad”, och bilden blir rörig ända tills nästa fullständiga bildruta tas emot. För att få en mjuk snabbspolning är det nödvändigt att söka upp nästa fullständiga ruta och hoppa dit. Det kommer att krävas att man förstår formatet för videoströmmen, och kan bli nödvändigt att i förväg indexera videomaterialet för att veta var dessa bildrutor finns. Detta är vad som skall undersökas närmare i detta diplomarbete – är det möjligt att implementera en funktionalitet som gör denna snabbspolning på ett bättre sätt? Hur kommer strömmande media över HTTP att fungera om ett paket tappas bort och TCP sänder det igen?

Utöver en funktion för felfri förflyttning i videoströmmen önskar man kunna förflytta uppspelningen med sekunder istället för antal tecken. Eftersom MPEG-formatet inte har konstant bitflöde kommer det att bli nödvändigt att läsa in tidsreferenser ur filen, för att sedan räkna ut var uppspelningen skall fortsätta.

Hibox Systems har införskaffat utvecklingsverktyg till Kreatels mottagare som möjliggör att delvis ändra dess funktioner. Det är enkelt att sätta till vad som skall hända när man under uppspelning trycker på knapparna för snabbspolning; i detta fall att vidarebefordra en förfrågan till servern. Frågor kvarstår, bland annat finns det en buffert i mottagaren eller videospelarprogrammet som gör att det tar över två sekunder innan en ändring i serverns utsändning av videoströmmen syns på uppspelningen. Man har redan en funktion för att pausa uppspelningen, och med den kan man tydligt se att det tar några sekunder att tömma bufferten. Om man kunde tömma denna buffert på något sätt skulle uppspelningen bli smidigare.

Kapitel 2

Format och protokoll för strömmande media

2.1 MPEG-standarderna

2.1.1 Historia och bakgrund

MPEG (Moving Picture Experts Group) är mera än bara en samling videoformat. Det är en sammanslutning av sakkunniga och intressenter som utformar standarder för komprimering av video och audio. I MPEGs bakgrundsinformation på den officiella hemsidan [22] nämns att gruppen ursprungligen grundades i januari 1988 och från början bestod av 25 medlemmar. Gruppen håller möten ungefär tre gånger per år enligt behov, och ett möte pågår i ungefär en vecka. Dessa konferenser hålls på varierande orter i världen och har idag ca 350 deltagare, både från kommersiella parter och från ideella organisationer och universitet. MPEGs officiella benämning är WG11 (Working Group 11) i SC29 (Sub-Committee 29), i samarbetet "Joint ISO/IEC Technical Committee (JTC 1) on Information Technology" [22].

MPEG har producerat ett antal standarder. Gruppens officiella hemsida [22] nämner de mest relevanta som definierar video och audio: MPEG-1, MPEG-2 och MPEG-4, vars ursprungliga förslag till respektive standard framlades år 1989, 1991 och 1995. Dessa jämförs av John Watkinson i "The MPEG Handbook" [23]. MPEG-1, som utvecklades för att användas på VideoCD-formatet, var ursprungligen baserat på en bithastighet av 1.5 MBit/s. Med samma kapacitet och speltid som en okomprimerad audio-CD kunde man med MPEG-1 få plats med både ljud och bild. MPEG-1 är gjort för att ha en upplösning på 352*288 bildpunkter med 25 bilder per sekund, respektive 352*240 punkter med 30 bilder per sekund. Dessa s.k. CIF-format (Common Intermediate Format) är hälften av PAL eller NTSC, respektive. MPEG-1 är dock inte begränsat till dessa upplösningar, utan ända upp till 4095*4095 understöds [24].

MPEG-2 är en utökning av MPEG-1. Från att ha varit ett enkelt men dock något begränsat format ger MPEG-2 många nya funktioner. Watkinson [23] betonar bakåtkompatibiliteten, en MPEG-2-avkodare[1] klarar av allt MPEG-1-material. En av de funktioner som lagts till jämfört med MPEG-1 är stöd för interlacing eller "radsprång", video som i likhet med en vanlig TV-sändning ritar varannan linje per bildruta. MPEG-2 består av fyra standardupplösningar, "nivåer", samt sex olika "profiler" för skilda användningsområden – exempelvis bandbredd, kvalitet och bakåtkompatibilitet. MPEG-2 kommer att behandlas mera ingående senare.

MPEG-4 är den senaste inkarnationen av standarden, och existerar i versionerna 1 och 2. Watkinson [23] behandlar de nya metoderna som används för att beskriva rörelser, på mycket mera dynamiska sätt än i MPEG-1 och MPEG-2. MPEG-4 introducerar möjligheter att isolera rörliga objekt från bakgrunden och återanvända bakgrunden till efterföljande bildrutor. De rörliga objekten kan, förutom att flyttas och roteras tvådimensionellt över bilden, även töjas och vridas tredimensionellt. På det sättet kan rörelser beskrivas mera effektivt än med fasta makroblock som används i MPEG-1 och MPEG-2. Watkinson [23] beskriver vidare funktionerna for att animera ansikten och figurer. MPEG-4 möjliggör tack vare detta t.ex. videokonferens över väldigt låg bandbredd. MPEG-4 tillämpar, i likhet med MPEG-2, ett antal olika profiler för olika användningsområden. Vissa av dem är utformade för verkligt filmmaterial, medan andra är gjorda för datoranimerat material.

2.1.1.1 Standardiserad bitström

Watkinson [25] beskriver en av de kanske största principerna för MPEG – ingenting i standarden säger hur en MPEG-enkoder[2] skall vara utformad. Vad som däremot är noggrant specificerat är hur en avkodare skall fungera. Detta antas bana väg för bättre MPEG-enkoder som mera effektivt utnyttjar formatets potential samt för mera konkurrens mellan kommersiella MPEG-enkodare. Avkodaren kan göras förhållandevis enkel eftersom den endast behöver kunna tolka en välkänd bitström, medan enkodern tillåts göra avancerade beslut om enkodningen så länge bitströmmen ger korrekt resultat ur en kompatibel avkodare.

2.1.2 Generell videokomprimeringsterminologi

MPEG bygger på s.k. förlustkomprimering, vilket innebär att man tappar (slänger) bort information vid komprimeringen. Denna information går aldrig att återskapa, till skillnad från vid förlustfri komprimering från vilken man alltid får tillbaka originalmaterialet.

[1] funktionalitet som återger MPEG-material, ofta visuellt och audiellt

[2] funktionalitet för att komprimera video/audiomaterial till MPEG-format

Därför är det viktigt att man väljer att lämna bort informationen på ett sådant sätt att det inte märks nämnvärt på slutresultatet.

Watkinson [26] beskriver hur det mänskliga ögat uppfattar bilden och vad vi generellt missar; som man kan förvänta sig finns en övre gräns på upplösningen som ligger på ca 50 cykler per grad (vinkel utgående från ögat, oberoende av avstånd). Vidare har synsinnet en viss tröghet, som gör att det tar från 0.15 till 0.3 sekunder att "läsa in" en bild. Även om man har ett begränsat antal receptorer (tappar och stavar) rör sig ögat fram och tillbaka hela tiden (saccadisk ögonrörelse), och ökar på så sätt upplösningen. För att hjärnan skall kunna kombinera dessa inlästa delbilder hålls de en stund i minnet, och det gör att synsinnet har en gräns i tidsperspektiv för hur snabba rörelser som går att uppfatta. Detta gör att man inte uppfattar flimmer från ljuskällor vars frekvens är tillräckligt hög – detta kallas den kritiska flimmerfrekvensen och ligger omkring 40-70 Hz beroende på ljusets intensitet. Vidare uppfattar man inte mycket detaljer av objekt som rör sig snabbt, men man följer instinktivt rörelse vilket gör att man uppfattar mindre detaljer i bildens stillastående detaljer. Dock är inte sagt att det räcker till att ha en bilduppdateringsfrekvens över den kritiska flimmerfrekvensen – ögat ser på olika delar av skärmen hela tiden och det leder till att man märker av blinkningar, s.k. bakgrundsblink.

Vidare behandlar Watkinson [26] hur små kontraster, eller skillnader i intensitet, ögat kan uppfatta. Detta är olinjärt beroende på ljusets intensitet, och i stort sett alltid konstant 1% av intensiteten. En katodstråleskärm svarar inte linjärt på inspänning, vilket innebär att man måste förstärka signalen olinjärt med en s.k. inverterad gamma-funktion. Från det faktum att ögats känslighet är beroende av ljusets intensitet får man att brus syns tydligare i mörka delar av bilden. Genom att göra den inverterade gamma-förstärkningen direkt vid kameran kan man undvika mycket linjebrus. Ögats kontrastkänslighet på större vinklar är mindre, vilket innebär att det är viktigt att bibehålla samma medelintensitet med närliggande punkter.

Färgseendet är individuellt, men generellt kan sägas att det mänskliga synsinnet är mest känsligt för grönt, därefter rött och minst känsligt för blått. Eftersom vi har tre olika receptorer som är känsliga för vardera grundfärg är det naturligt att kameror också registrerar dessa färger skilt för sig. Däremot är det inte effektivt utnyttjande av bandbredd att skicka dessa signaler var för sig. Watkinson [26] visar formel 2.1 för att beräkna ett viktat luminansvärde Y utifrån R-, G- och B-värden:

$$Y = 0.3R + 0.59G + 0.11B \tag{2.1}$$

Denna monokroma signal motsvarar ungefär ögats uppfattning av vardera färgerna. De andra färgerna kan då överföras som skillnader från denna. Endast två färgdifferenssignaler behövs eftersom den tredje kan räknas ut. Eftersom gröna komponenten redan utgörs

så stor del av luminansen Y, överförs i allmänhet färgskillnaderna R-Y och B-Y. Detta är en av de tidigaste formerna av videokomprimering, och utöver att spara bandbredd ger detta i analoga TV-system bakåtkompatibilitet med svartvita kameror och mottagare.

En annan tidig form av komprimering som Watkinson [26] nämner är radsprång, eller interlacing, vilket innebär att man ritar udda/jämna horisontella linjer varannan gång. Detta används sedan första början i båda de analoga TV-systemen PAL och NTSC. Här utnyttjas synsinnets tidsmässiga tröghet, och på så sätt kan bandbredden halveras – eller bildfrekvensen alternativt upplösningen fördubblas. Radsprång har dock nackdelar – vid sekvenser med objekt som rör sig snabbt kommer kanterna av detta att bli suddigare än vid progressiv video (som ritar varje linje per bildruta), och om man skulle pausa uppspelningen uppstår en “kam” av de udda/jämna linjerna.

2.1.3 Videokomprimering med MPEG

Vid videokomprimering med MPEG försöker man till stor utsträckning eliminera överflödig (redundant) information i materialet. Man har dels en redundans inom varje bildruta – exempelvis stora ytor med samma färg – dels en redundans mellan bildrutor – många detaljer hinner sällan ändras mellan två rutor. Genom att göra sig av med onödig information åstadkoms en besparing av bandbredd och lagringsutrymme.

2.1.3.1 Enskild bildruta – blockindelning och diskret cosinustransform

För komprimeringen av enskilda bildrutor i MPEG-2 beskrivs av Watkinson [27] den diskreta cosinustransformen (DCT). MPEG-4 introducerar en ny metod med krusningar (wavelets). En enskild komplett bildruta benämns Intra-frame, eller I-frame. Den diskreta cosinustransformen innebär att man överför materialet till frekvensplanet. Därmed möjliggörs att plocka ut de mest signifikanta frekvenserna, vilket gör att DCT är en central del i MPEG.

Eftersom DCT är baserad på block av data delas bilden upp i ett antal kvadratiska områden. MPEG använder 8x8 pixlar som grundläggande DCT-block, samt 16x16 (fyra DCT-block) för ett s.k. makroblock. Watkinson [28] beskriver MPEG-strukturen, dvs. m:n:o (t.ex. 4:2:0) i makroblocken, där m anger antalet luminansblock, n anger horisontell nersamplingsfaktor (medelvärde av närliggande) av färgdifferensblocken i förhållande till luminansblocken, och o är samma som n, förutom när o är noll då färgdifferensblocken nersamplas vertikalt med en faktor av två. 4:2:0 innehåller fyra luminansblock, ett R-Y- samt ett B-Y-block, 4:2:2 innehåller likaså fyra luminansblock men två R-Y- och två B-Y-block. Högsta kvalitet får man naturligtvis med 4:4:4, som innebär att man inte nersamplar färgkanalerna överhuvudtaget.

Dessa block transformeras var för sig med en tvådimensionell diskret cosinustrans-

form till frekvensplanet. Reimers [29] visar ekvation 2.2 som används i MPEG för en 8x8-punkts DCT.

$$G(f_x, f_y) = \frac{1}{4}C(f_x)C(f_y)\sum_{x=0}^{7}\sum_{y=0}^{7} g(x,y) * cos((2x+1)f_x\frac{\pi}{16}) * cos((2y+1)f_y\frac{\pi}{16}) \quad (2.2)$$

där $C(n) = \frac{1}{\sqrt{2}}$ om $n = 0$
$C(n) = 1$ om $n > 0$
samt
f_x, f_y = frekvenskoordinater i x- respektive y-led,
$G(f_x, f_y)$ = DCT-koefficient för frekvenskoordinaten i fråga,
x,y = pixelns koordinater i blocket,
$g(x,y)$ = pixelns värde i koordinaten x,y.

Detta ger en 8x8-matris med frekvenskoefficienter, från DC-komponenten uppe i vänstra hörnet ($x,y = 0,0$) till den högsta frekvensen nere i högra hörnet($x,y = 7,7$). Fortfarande har ingen komprimering skett, utan värdena måste först kvantiseras. Här divideras vardera koefficient med en kvantiseringskoefficient och avrundas till heltal. Kvantiseringskoefficienterna som Reimers [29] beskriver, är en annan 8x8-matris med konstanter som är olika för luminans och krominans. Dessa matriser har noggrant valts ut för att ge så liten märkbar kvalitetsförsämring som möjligt.

Efter kvantisering kommer i allmänhet de flesta värden att vara samlade uppe i vänstra hörnet av matrisen, medan de högre frekvensvärdena har kvantiserats till noll. Därefter tas de ut i ett sicksackmönster från DC-komponent till högsta existerande frekvenskoefficient, och därefter sätts en EOB-symbol (End Of Block) in för att indikera att resten av elementen är noll. Utöver detta görs en Huffman-enkodning för att mera effektivt beskriva upprepade sekvenser.

Av dessa 8x8-block görs ett större 16x16-makroblock, och slutligen hela bildens upplösning, t.ex. 720x576 som består av 45x36 st makroblock. Detta är vad som kallas en komplett I-ruta. Watkinson [30] tar upp konceptet med remsor, eller slices, som är en rad av efterföljande makroblock från vänster till höger. Här kan man utnyttja det faktum att den genomsnittliga luminansen, eller DC-koefficienten, från kvantiseringen ofta är samma i många efterföljande makroblock. Den sänds i början av en remsa, och om den ändras sänds bara skillnaden. Remsorna innehåller synkroniseringsinformation som indikerar var på rutan den skall börja, vilket ökar säkerheten vid överföringsfel.

2.1.3.2 Komprimering mellan flera bildrutor – P- och B-rutor

Även om I-rutor komprimerar bild-datan markant är många på varandra följande rutor ofta väldigt lika – detta är vad som kallas temporal redundans – och det är därför onödigt att sända samma bild-data en gång till. MPEG gör det möjligt att överföra bara skillnaderna mellan efterföljande rutor.

Watkinson [30] beskriver ett antal metoder som används i MPEG för att beskriva förändringar mellan efterföljande bildrutor. Utöver att helt enkelt sända subtraherade skillnaderna mellan två rutor tillämpar MPEG en metod med rörelsevektorer för att beskriva rörelser i materialet. Här sänds information om hur enskilda makroblock skall flyttas vertikalt och horisontellt i nästa bildruta, och på så sätt effektiveras komprimeringen. Även om denna rörelsekompensering ger en grov beskrivning av händelserna måste man ändå skicka över en del korrigerande differensdata, som avkodaren lägger till efter att ha flyttat makroblocken. Detta är vad som bygger upp en s.k. predikterad ruta, eller P-frame. Dessa P-rutor kan även innehålla kompletta I-makroblock, även om man i allmänhet bara använder rörelsevektorer och predikteringsfel på P-rutor.

En tredje typ av bildrutor är tvåvägs/bidirektionell enkodning, s.k. B-frames. Watkinson [30] visar ett exempel med ett scenario när ett rörligt objekt frilägger en ny bakgrund. Detta kunde visserligen beskrivas med I-makroblock, men B-rutor möjliggör att skapa det från följande och/eller föregående P- eller I-ruta. B-rutor predikteras endast från I- eller P-rutor, inte från närliggande B-rutor. Sällan använder man mera än två B-rutor i följd, eftersom interpolationen skulle ge för stora fel.

Vinsten med dessa temporala redundanselimineringar är stor. Röjne [31] nämner några typiska värden på 30% i utrymmesbehov för P-rutor och 12% för B-rutor, i förhållande till I-rutor. Dock måste man komma ihåg att P- och B-rutor tenderar att vidarebefordra fel så denna inbesparning kommer på bekostnad av bildkvaliteten. Det är också nödvändigt att starta avkodningen från en komplett I-ruta.

Group Of Pictures (GOP): När enkodern genererar en bitström bestående av I-, P- och B-rutor används dessa i en bestämd sekvens – en grupp av rutor eller Group Of Pictures. Watkinson [30] ger sekvensen IBBPBBPBBPBB som exempel på en vanlig struktur, och en GOP börjar alltid med en I-ruta. Det kan vara önskvärt att varje GOP skall vara oberoende av tidigare och efterföljande rutor, och då måste man antingen göra den sista B-rutan bakåtpredikterad, eller använda en sekvens med en P-ruta i slutet – detta kallas en sluten GOP.

När man sänder B-rutor som skall rekonstrueras från framtida P- eller I-rutor t.ex. över nätverk ställs man inför problemet att denna information inte ännu finns tillgänglig hos mottagaren. En möjligt lösning kunde vara att ha en tillräckligt stor buffert på mottagarens sida, men Watkinson [30] presenterar MPEGs lösning – man ändrar på rutornas ordning

så att en sekvens som $I_1B_2B_3P_4B_5B_6P_7B_8B_9I_{10}B_{11}B_{12}P_{13}$ blir
$I_1P_4B_2B_3P_7B_5B_6I_{10}B_8B_9P_{13}B_{11}B_{12}$, m.a.o. sänds I- och P-rutor före de B-rutor som är beroende av dem. Rutorna har naturligtvis tidsinformation för att avkodaren skall kunna sätta dem i rätt ordning.

2.1.4 Audiokomprimering i MPEG

Audio komprimeras med liknande metoder som rörliga bilder – man gör sig av med onödig information till den grad att slutresultatet fortfarande håller en acceptabel kvalitet. I likhet med synsinnet kan människans hörsel luras att tro att ingen information saknas. Watkinson [32] beskriver hörselns uppfattning av ljudnivåer, den s.k. phon-skalan, som genom psykoakustiska test har utvecklats för att visa hur låga tryckförändringar örat kan uppfatta. Känsligheten beror främst på ljudets frekvens, men är också individuellt och försämras med åldern. Känsligast är man dock för ljud mellan 200 Hz och 5 kHz, även om hörseln kan uppfatta svängningar mellan 20 Hz och 20 kHz.

Den kanske viktigaste delen som Watkinson [32] tar upp är förmågan att urskilja enskilda frekvenser. På grund av hörselcellernas, de s.k. hårcellernas, anatomi med ett hårliknande membran som vibrerar till följd av ljudvågor, kommer en dominerande frekvenskomponent att maskera svagare ljud. Vidare har dessa hår en tröghet som gör att det tar en stund för dem att börja vibrera, vilket gör att korta ljudpulser upplevs svagare även om ljudtrycket är förhållandevis högt. Tillsammans ger dessa egenskaper en s.k. kritisk bandbredd på ca $\frac{1}{3}$ oktav, under vilken man inte kan särskilja två frekvenser. Om det existerar flera närliggande kraftiga frekvenskomponenter kan dessa blandas ihop till en och samma signal, som pulserar med en frekvens, vilken är differensen av de båda signalernas frekvenser.

Dessa brister i hörselsinnet kan utnyttjas för att reducera utrymmesbehovet för digital audio. Watkinson [33] behandlar de tre grundläggande audiolagren i MPEG, lager I, II och III. Ursprungligen härstammar de tre nivåerna från två olika projekt, dels det europeiska MUSICAM (Masking pattern adapted Universal Sub-band Integrated Coding And Multiplexing), dels det huvudsakligen amerikanska ASPEC (Adaptive Spectral Perceptual Entropy Coding). De olika lagren bygger på varandra och högre lager ger större funktionalitet för att ge effektivare komprimering med bibehållen kvalitet, med krav på högre beräkningskapacitet och minne. Bithastigheten kan väljas i ett antal steg mellan 32 och 384 kbps, och ett antal kanalkonfigurationer som mono, dubbelmono, stereo och förenad stereo understöds. Alla nivåer stöder dock inte alla bithastigheter.

MPEG lager I är den enklaste nivån både att enkoda och avkoda. Watkinson [33] beskriver hur audiospektret från en pulskodmodulerad (PCM) signal delas in i 32 lika stora delar, i vilka ordlängden kan justeras beroende på materialet. En 512-punkts fouri-

ertransform används parallellt med bandindelningen för att analysera insignalens spektrum. Från detta spektrum kan identifieras de dominerande frekvenskomponenterna, och hur hög brusnivån är för att kunna veta hur kort ordlängd som kan användas för att kvantisera vardera av de 32 delbanden. Dessa block resulterar i den s.k. elementära strömmen. Avkodaren som Watkinson [33] beskriver fungerar i stort sett på motsatt sätt, genom att återigen skala delbanden enligt skalningsfaktorn, för att slutligen inversfiltrera dem och återfå det något som påminner om det ursprungliga spektret.

MPEG lager II bygger vidare på lager I men Watkinson [33] pekar dock på några förändringar – storleken på insignalblocken har tredubblats från 384 till 1152 samplingspunkter, och den parallella fouriertransformens upplösning har fördubblats till 1024. Vad gäller kvantiseringen har lager II en klassificering mellan låg-, mellan- och högfrekvent register som vardera kvantiseras med 15, sju och tre olika ordlängder respektive. Lager II besparar bandbredd genom att inte sända skalningsfaktorer förutom när de ändras. Det är lager II som ofta används i bl.a. DVB och DVD.

MPEG lager III, allmänt känt som MP3, utökar lager II med ett antal avancerade funktioner. De största nyheterna Watkinson [33] beskriver är dels de överlappande fönsterfunktionerna och en förbättrad psykoakustisk modell. De överlappande fönsterfunktionerna är en metod för att förkorta samplingar vid transienter, med normal storlek på 36 samplingar och förkortas till tolv samplingar vid transient material. Detta möjliggör att bättre transientåtergivning men färre frekvenser kan återges.

Vidare kan nämnas MPEG-2 AAC (Advanced Audio Coding) som ger en samling nya audiokodningsmetoder. AAC är orienterat mot hög kompression men god kvalitet och realistiskt ljud. MPEG-4 bygger vidare på AAC med ännu mera avancerade funtioner såsom mottagargenererat brus istället för att försöka komprimera bruset vilket är ineffektivt. Slutligen nämner Watkinson [33] Dolby AC-3 som också det påminner mycket om AAC och MPEG lager III.

2.1.5 MPEG-protokollet i DVB – MPEG-2 (ISO/IEC 13818)

Det videoformat som ursprungligen valts för DVB är MPEG-2 eller ISO/IEC 13818. Enligt ETSIs DVB-specifikationer [34] understöds det i såväl bredbild (16:9) som normal (4:3). Både 25 Hz och 30 Hz uppdateringsfrekvens kan användas. Ljudet kan sändas som MPEG-1 lager 2 i ett antal olika kanalkonfigurationer. Dessa är vad som huvudsakligen används idag (2006). Vidare finns specificerat stöd för H.264 (MPEG-4 Part 10, AVC (Advanced Video Coding)) och HE-AAC (High-Efficiency Advanced Audio Coding, MPEG-4 Part 3) som audioformat. Alla dessa videoformat har specificerats både i normal- och HDTV-upplösning.

MPEG-2 är vad som kommer att behandlas ingående här.

2.1.5.1 Profiler och nivåer

MPEG-2-standarden definierar ett antal profiler och nivåer, uppräknade i tabell 2.1, för att ha gemensamma standarder för olika användningsområden. Dessa beskrivs av Watkinson [23] som att högre profil ger tillgång till mera avancerade komprimeringsfunktioner, men kräver samtidigt mera beräkningskapacitet och minne, vilket kan ha betydelse i t.ex. mobila applikationer. Här kan Simple-profilen vara användbar, eftersom den saknar B-rutor. Main-profilen är vad som oftast används, bland annat i DVB och DVD. Nivåerna är ett antal fördefinierade upplösningar, och det är Main-nivån på 720x576 punkter som används i DVB.

	Simple	Main	4:2:2	SNR	Spatial	High
High		1920x1152 90 Mbps				1920x1152 100 Mbps
High 1440		1440x1152 60 Mbps			1440x1152 60 Mbps	1440x1152 80 Mbps
Main	720x576 15 Mbps	720x576 15 Mbps	720x576 50 Mbps	720x576 15 Mbps		720x576 20 Mbps
Low		352x288 4 Mbps		352x288 4 Mbps		

Tabell 2.1: MPEG-2-formatets definierade profiler och nivåer tillsammans med ungefärligt krav på bithastighet, Watkinson [23], s. 20

SNR och Spatial-profilerna är såtillvida intressanta, att de är skalbara genom att ha en tilläggssignal som, om mottagaren har tillräckliga resurser, kan användas för att öka bildkvaliteten. SNR-profilens tilläggssignal strävar till att förbättra bildens signal-brusförhållande som uppkommer som följd av kvantiseringsfelet i enkodern, medan Spatial-profilens tilläggssignal höjer upplösningen från SDTV- till HDTV-nivå. Eftersom profilerna delvis utökar varandra har High-profilen stöd för både SNRs och Spatials funktionalitet. 4:2:2-profilen är den enda som har en annan krominans-nersampling är 4:2:0. 4:2:2 används huvudsakligen under produktion och redigering.

2.1.5.2 Tre lager av strömmar

När video- och audioenkodern producerar var sin ström av data måste de paketeras på lämpligt sätt för att kunna överföras tillsammans. Watkinson [35] beskriver ytligt dessa i kapitlet “Program and transport streams”. En elementär bitström (ES, Elementary Stream) är vad enkodern ger ut. Här finns de grupper av bilder (GOP) som tidigare behandlats. Nästa nivå är den paketerade elementära strömmen (PES, Packetized Elementary Stream) som delar upp de elementära strömmarna i hanterliga block tillsammans med tidsinformation. Slutligen det översta lagret, transportström (TS, Transport Stream), som möjliggör samtidig överföring av flera program bestående av video, audio och data.

Den elementära videoströmmen, ES: Formatet för den elementära videoströmmen beskrivs av ISO/IEC i standarden 13818-2 [36] med startkoderna och prefixen som indikerar början på bl.a. rutor, remsor och GOP. Alla börjar med startkodsprefixet 0x000001, därefter följer den egentliga startkoden, enligt tabell 2.2.

Typ	Kod, hexadecimalt
Bild	0x00
Remsa	0x01–0xAF
Reserverad	0xB0, 0xB1
Användardata	0xB2
Sekvens-header	0xB3
Sekvensfel	0xB4
Utökad startkod	0xB5
Reserverad	0xB6
Sekvens slut	0xB7
Grupp-startkod	0xB8
Systemstartkod (PES, PS)	0xB9–0xFF

Tabell 2.2: Videoströmmens startkoder, ISO/IEC 13818-2 [36], s.20

Flera av dessa har utökade startfält med mera information men kommer inte att behandlas ingående här. Samma headerstruktur delas med PES och PS (behandlas senare), som använder systemstartkoderna bland annat för att indikera om materialet är video eller audio. Den kod som kanske är mest intressant i detta fall är grupp-startkoden, dvs. 0x000001B8, som indikerar start av en ny grupp av bilder eller GOP, samt bild-startkoden, 0x00000100.

Paketerad elementär videoström, PES: För att enklare kunna hanteras delas den elementära strömmen upp i en paketerad elementär ström (PES, Packetized Elementary Stream). Storleken på ett PES-paket är ganska fri, och paketstrukturens längdfält är 16 bitar, vilket betyder maximalt 65535 tecken. Enligt Röjne [37] brukar dock ett PES-paket innehålla en bildruta, eller 24 ms ljud.

ISO/IEC 13818-1 [38] definierar att ett PES-paket börjar med samma prefix, 0x000001, precis som den elementära bitströmmens komponenter. Därefter följer en ström-ID på ett tecken, som tolkas enligt tabell 2.3.

Därefter följer ett längdfält på två tecken. Ur standarden följer att detta definierar innehållets längd börjande efter det andra tecknet. Längdvärde noll kan tillåtas om videomaterialet skall paketeras i en transportström. Efter längdfältet kommer vidare bitfält med information om bl.a. kryptering och tidssynkronisering.

Tidssynkroniseringsmekanismen i den paketerade elementära strömmen, beskriven av Watkinson [35], består av PTS-fält (Presentation Time Stamp) och DTS-fält (Decode Time Stamp). PTS används av både video- och audioströmmar, medan DTS endast används i videoströmmar för att beskriva var t.ex. en B-ruta skall avkodas. Beroende på överfö-

Kod, hexadecimalt	Typ
0xBC	Strömkarta/Program Stream Map
0xBD	Privat ström 1
0xBE	Utfyllnad
0xBF	Privat ström 2
0xC0–0xDF	Audioström, MPEG-1/MPEG-2/MPEG-4 (ISO/IEC 11172-3/13818-3/13818-7/14496-3)
0xE0–0xEF	Videoström, H.262 MPEG-1/MPEG-2/MPEG-4 (ISO/IEC 11172-2/13818-2/14496-2)
0xF0	ECM-ström
0xF1	EMM-ström
0xF2	DSM-CC (ISO/IEC 13818-1/13818-6)
0xF3	ISO/IEC 13522 (interaktivitet/hypermedia)
0xF4	ITU-T H.222 typ A
0xF5	ITU-T H.222 typ B
0xF6	ITU-T H.222 typ C
0xF7	ITU-T H.222 typ D
0xF8	ITU-T H.222 typ E
0xF9	Underordnad ström
0xFA–0xFE	Reserverad
0xFF	Programströmkatalog/Program Stream Directory

Tabell 2.3: PES ström-ID, ISO/IEC 13818-1 [38], s.34

ringens natur sätter man in dem med olika stora mellanrum, från ca 100 ms på en osäker transportström upp till 700 ms på ett välkänt feltolerant medium som en programström på DVD. Watkinson [35] beskriver att dessa 33-bitars tidssynkroniseringsfält härstammar ur MPEG-enkodern, från värdet av en räknare som inkrementeras av en klockpuls på 90 kHz. Denna klockpuls har i sin tur dividerats ner från en klocka på 27 MHz. Om ljudet och videon komprimerats enligt samma tidsreferens kan de med hjälp av dessa fält synkroniseras i mottagaren.

Slutlig multiplexing samt paketering – transportström (TS) och programström (PS): Slutligen kan dessa PES paketeras i en transportström, TS. Transportströmmen kan innehålla många olika paketerade elementära strömmar, och t.ex. i Finland används för närvarande endast tre transportströmmar (”kanalknippen” eller multiplexar) för digital-TV (Digita [39]). Watkinson [35] beskriver denna multiplexning, där alla strömmars PES med jämna mellanrum tillåts sändas ut för att få en rättvis fördelning av bandbredden. Varje TS-paket har ett identifikationsnummer på 13 bitar, PID eller Packet Identification Code, som gör att man kan hålla reda på vilka TS-paket som innehåller vilken PES-ström. TS-paketen är alltid 188 tecken, vilket betyder att de större PES-paketen ofta sträcker sig över flera TS-paket. Konstant storlek gör felkontroll enklare, samt att demultiplexern enklare kan välja ut vilka paket den behöver. En annan viktig egenskap hos transportströmmen i

DVB är att bithastigheten är konstant – om materialet är mindre än det angivna kommer nollpaket med PID 8191 att läggas in.

Fält	Bitar
Synkroniseringstecken (0x47)	8
Transportfelsindikator	1
Startindikator för nyttolast	1
Prioritet	1
PID	13
Kryptering	2
Anpassningsfält (utökad header)	2
Löpnummer (per PID)	4

Tabell 2.4: Format för transportströmpaketets headerstruktur, ISO/IEC 13818-1 [40], s. 18-20

ISO 13818-1 [40] definierar formatet för transportströmpaketets huvud uppräknat i tabell 2.4. Synkroniseringstecknet är alltid samma (01000111), transportfelsindikatorn kommer att vara 1 om ett fel upptäckts någonstans i överföringen. Startindikatorn för nyttolast satt till 1 betyder att TS-paketet börjar med ett nytt PES-paket. Prioriteten kan sättas för att ge paket högre prioritet, och PID har redan behandlats. Krypteringsindikatorn visar om strömmen är krypterad, och ett värde på 00 betyder att den är okrypterad. Anpassningsfältets indikator visar om en utökad header följer efter egentliga headern. Detta kan förutom att överföra mera information (bland annat klockreferens) användas för att fylla ut TS-paketet om resterande del av ett PES-paket inte når upp till 184 tecken. Anpassningsfältet kommer att behandlas mera ingående senare. Slutligen ett löpnummer för de paket som hör till en viss PID, inkrementeras för varje paket och börjar sedan om från noll.

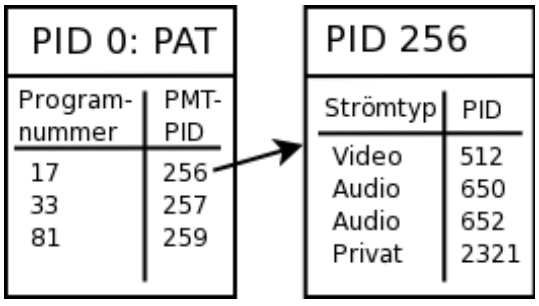
Utöver detta behandlar Watkinson [35] systemet för att överföra programinformation – Program Specific Information eller PSI. Det består av den s.k. Program Association Table eller PAT, som innehåller en lista över tillgängliga program (som består av en videoström och en eller flera audioströmmar, samt undertexter) tillsammans med PID för deras Program Map Table eller PMT. PMT listar i sin tur de PID för video- och audioströmmar som hör till programströmmen i fråga. PSI är en viktig del av DVB och kommer att behandlas ingående senare.

Ett annat sätt att paketera video tillsammans med audio är en programström. Röjne [37] nämner DVD som användningsområde för denna typ av multiplex, och programströmmen lämpar sig för användningsområden där risken för överföringsfel är förhållandevis låg. Programströmmen använder hela PES-paket, och paketstorleken är därmed varierande. ISO/IEC 13818-1 [41] visar hur strömmen är uppbyggd av s.k. packs, en kedja av ett antal PES-paket av både video- och audiotyp tillsammans med tidsinformation. Ett pack börjar precis som ett PES-paket med prefixet 0x000001 och sedan tecknet 0xBA,

därefter följer tidsinformation samt information om strömmens innehåll. Slutligen termineras alltid en programström med sekvensen 0x000001B9.

Programspecifik information, PSI: Eftersom transportströmmen består av många olika programhelheter som i sin tur består av en videoström och oftast minst en audioström, samt i allmänhet text-TV- och MHP-strömmar finns behov av ett system för att hålla reda på vilka strömmar som skall tas emot. För detta har ISO/IEC 13818-1 [42] ett standardiserat system för att överföra tabeller som beskriver programmen, Program Specific Information eller PSI.

Det första steget är att leta upp PAT (Program Association Table), som alltid hittas i ett transportpaket med PID noll. ISO/IEC 13818-1 [42] beskriver, att man från dessa paket kan läsa in en lista av de program som transportströmmen består av. Programnumret, ett heltal, listas tillsammans med en PID för programmets PMT (Program Map Table). Programnummer noll ger PID för nätverksinformationstabellen NIT (Network Information Table) som innehåller information om hur distributionsnätet är uppbyggt och bl.a. vilka andra transportströmmar (och deras frekvenser) som finns tillgängliga. ISO 13818-1 [40] tillåter PID 16 till 8190 för valfri ström, och till dem hör utöver mediaströmmarna även PMT. PMT listar de elementära strömmar som bygger upp programmet tillsammans med en beskrivning av deras innehåll, och principen finns illustrerad i figur 2.1. Båda typerna av tabeller är snarlika och identifieras förutom av PID med fältet table_id, som för PAT är noll och för PMT två. Tabellerna kan sträcka sig över flera transportpaket, vilket innebär att man måste hålla reda på hur mycket data som finns kvar att läsa in. Transportströmpaket som innehåller början på en ny tabell kommer att ha payload_start-flaggan satt till ett.



Figur 2.1: Principen för PSI – PAT ger referens till de enskilda programmens PMT-tabeller

Tidssynkronisering och anpassningsfältet: Transportströmmen har ett avancerat system för tidssynkronisering som understöder PES-strömmens PTS/DTS-information. Watkinson [35] tar upp den centrala med PCR (Program Clock Reference). PCR används i avkodaren för att återskapa enkoderns klocka på 27 MHz, som sedan används vid syn-

kroniseringen av PES-paketen med hjälp av deras PTS/DTS-fält. En 48-bitars räknare inkrementeras av denna klocka och multiplexern läser med jämna mellanrum av räknarens värde. Detta värde sänds sedan ut i transportpaketets utökade pakethuvud, det s.k. anpassningsfältet (adaptation field). I mottagaren finns en s.k. faslåst slinga (phase-locked loop) som med hjälp av detta värde hålls synkroniserad. Alla strömmar behöver nödvändigtvis inte ha PCR, utan kan dela den med andra. ISO/IEC 13818-1 [42] definierar fältet PCR_PID i PMT som anger vilka transportpaket som innehåller denna information.

Anpassningsfältet existerar om detta finns indikerat i transportström-headerns avsedda fält; ISO/IEC 13818-1 [40] definierar att om detta är satt till två finns endast anpassningsfältet och om det är satt till tre finns både anpassningsfält och nyttolast.

Fält	Bitar
Längd	8
Diskontinuitetssindikator	1
Slumpmässig åtkomst	1
Prioritet för elementär ström	1
PCR	1
OPCR	1
Brytpunkt	1
Privat data	1
Utökning av anpassningsfält	1

Tabell 2.5: Format för anpassningsfältets headerstruktur, ISO/IEC 13818-1 [40], s. 20-30

ISO/IEC 13818-1 [40] beskriver dessa fält som står uppräknade i tabell 2.5; längden indikerar hur många tecken anpassningsfältet innehåller och kan vara noll om man endast behöver ett utfyllnadstecken. Diskontinuitetsindikatorn skall möjliggöra att nollställa avkodarens klocka vid hopp i transportströmmen, och är således synnerligen intressant i detta fall. Indikatorn för slumpmässig åtkomst är också intressant, eftersom den indikerar att transportströmpaketet i fråga innehåller information för att starta uppspelning därifrån. Prioritetsindikatorn kan användas för att visa att aktuellt paket skall ges högre prioritet. Därefter flaggor för att indikera att anpassningsfältet innehåller PCR och OPCR (Original Program Clock Reference). Brytpunktsindikatorn visar att anpassningsfältet innehåller en åtta-bitars räknare som anger antalet transportströmpaket tills ett paket med strukturer för slumpmässig åtkomst – även detta kan visa sig användbart i syfte att förflytta sig i transportströmmen. Indikatorn för privat data möjliggör överföring av implementationsspecifik data. Slutligen indikator för det utökade anpassningsfältet vars fält för att överföra nästa bildrutas DTS-data kan visa sig användbart.

2.1.5.3 Åtkomstkontroll och kryptering

Funktioner för att begränsa åtkomsten till materialet existerar både under transportströmnivån och den paketerade elementära strömmens nivå. Detta innebär att transportpaketen kommer att vara identifierbara även om innehållet är krypterat. De dominerande krypteringssystemen på DVB är Conax[3] samt Viaccess[4]. I finländska mark- och kabelsändningar används för närvarande endast Conax.

Röjne [37] beskriver i kapitlet "Transportströmmen" ytligt hur krypteringen av transportströmmen är uppbyggd. TS-paket med PID 1 är den s.k. Conditional Access Table, som innehåller information om vilka program som är krypterade. Vardera av dessa program har en egen s.k. Entitlement Management Messages-tabell, som beskriver hur programmet skall kunna avkodas. I övrigt är krypteringen av mediaströmmarna fri. Gemensamt för systemen är att de använder någon form av utbytbar sträng som utgör nyckel till avkodningen, och ofta en s.k. servicenyckel som byts med ungefär en månads mellanrum. Över denna ligger en huvudnyckel som behövs för att avkoda nya servicenycklar.

Kryptering på PES-nivå beskrivs kort av ISO 13818-1 [38] som indikerat av en tvåbitars flagga i ström-headern. Ett värde på noll betyder att strömmen är okrypterad. Vid kryptering förblir PES-headern intakt.

Åtkomstkontrollen realiseras i praktiken med ett s.k. smart card. Röjne [43] presenterar CA-modulen, som på en set-top-box antingen är inbyggd eller utgörs av ett PCMCIA-kort[5] som kan bytas ut enligt behov. PCMCIA-kortet eller den integrerade CA-modulen har i sin tur en kortläsare för ett smart card, vilket kunden erhåller vid avtalat abonnemang.

Under IPTV fungerar åtkomstkontrollen i stort sett på samma sätt som på DVB – transportströmmen är fortfarande samma men endast ett fåtal IPTV-mottagare har inbyggt Conax-gränssnitt.

2.1.5.4 Hopfogning och godtycklig åtkomst av strömmar

MPEG-standarden ISO 13818-1 [44] tar upp detta synnerligen relevanta ämne i tilläggskapitlet "Annex K: Splicing Transport Streams". Man behandlar hopfogning av transportströmmar som ursprungligen kan vara fullständigt orelaterade. I detta fall är det enklare på den punkten, att transportströmmen som man skall förflytta sig i, och i praktiken utföra en hopfogning av i realtid, är enhetlig och har samma upplösning, profil och bithastighet. ISO 13818-1 tar upp två olika sätt, sömlös (seamless) och icke-sömlös (non-seamless) hopfogning. Skillnaderna ligger i hur tidsinformationen förändras efter fogen; med sömlös hopfogning kommer tidssamplingarna att fortlöpa utan märkbar förändring. Icke-sömlös hopfogning i sin tur betyder att man tillåter en signifikant förändring i tidsinformationen,

[3] http://www.conax.com
[4] http://www.viaccess.com
[5] Personal Computer Memory Card International Association

vilket innebär en diskontinuitet. I ingetdera fall tillåts att tidsinformationen efter fogen är mindre än före.

För att kunna hitta lämpliga positioner för hopfogning beskriver ISO 13818-1 [44] brytpunktsflaggan i transportpaketets anpassningsfält. I fallet med endast brytpunktflaggan satt, innebär att transportpaketet efter paketet med flaggan innehåller strukturer för att göra en icke-sömlös hopfogning och fortsätta uppspelningen från – start av ny GOP. Följande paket kommer att kunna användas som början efter ett hopp. I fallet med sömlös brytpunkt kommer det utökade anpassningsfältet att innehålla DTS-fält för det efterföljande paketets nyttolast. Denna information tolkas dock inte av dekodern, utan skall användas vid hopfogning.

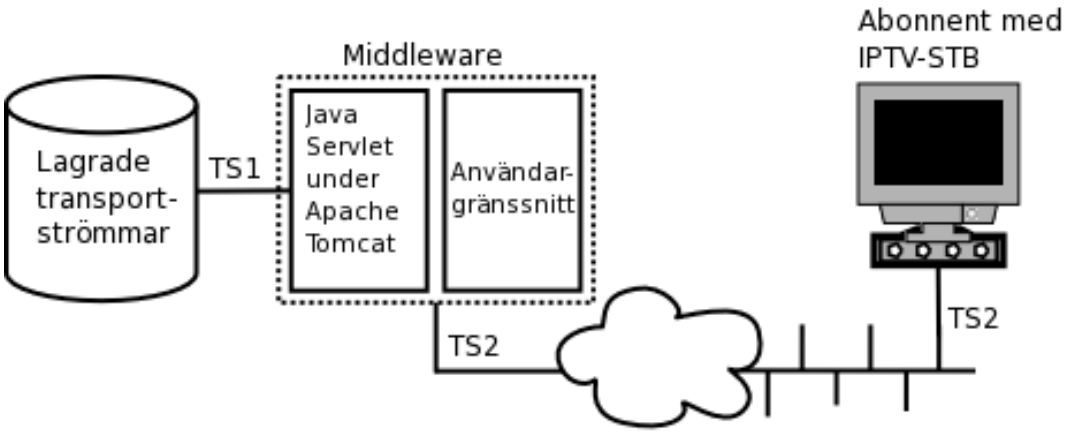
Vidare beskriver ISO 13818-1 [44] dekoderns beteende vid dessa brytpunkter. Vid icke-sömlös hopfogning skall efterföljande paket alltid innehålla PTS- och DTS-information. Det som kan ställa till problem i detta fall är behandlingen av diskontinuiteter i tidsinformationen, ett beteende som är beroende av dekodern. Enligt ISO 13818-1 är en vanlig metod att visa några frysta bildrutor. Med en sömlös hopfogning skall övergången däremot ske fullt transparent.

2.1.6 Möjliga tekniker för förflyttning och snabbspolning

Det tekniska problemet i detta fall är att man vill kunna förflytta uppspelningen framåt och bakåt i en MPEG-transportström över HTTP. Man har idag en implementation som hoppar ett fast steg framåt i transportströmmen men tar ingen som helst hänsyn till var uppspelningen fortsätter utan videoavkodningen kan fortsätta t.ex. mitt i datastrukturerna för ett PES-paket för audioströmmen. Denna information kan mottagarens MPEG-avkodare givetvis inte behandla, utan resultatet blir förvrängning av bilden med s.k. blockfel, med makroblock i tydligt felaktiga och onaturliga färger som fyller bilden tills nästa fullständiga bild har tagits emot. Mediaspelarprogrammet i Kreatel/Motorolas IPTV-mottagare har ibland även visat sig krascha fullständigt vid hopp i transportströmmen.

Man måste hålla skillnad på begreppen förflyttning och snabbspolning. Förflyttning i videoströmmen innebär att man fortsätter uppspelningen från godtycklig position, t.ex. ett tiotal sekunder framåt eller bakåt. Snabbspolning skulle spela upp videon snabbare än normalt, i likhet med en videobandspelare.

Hibox Systems PVR-funktion, illustrerad i figur 2.2, är sådan att när ett program spelas in lagras en transportström som en begränsad delmängd av den ursprungliga transportströmmen. Här ingår förutom videoströmmen och audioströmmar, alla text-, text-TV- samt MHP-strömmar som är associerade med programmet. Detta kan sägas vara en Single Program Transport Stream (SPTS). Denna transportström skrivs ner till disk och sänds sedan över HTTP till användaren. Detta är för övrigt samma transportström som sänds



Figur 2.2: Principskiss för IPTV-systemets PVR-funktion. En Java-servlet i middleware-programvaran levererar transportströmmen TS2 åt abonnentens set-top-box, från TS1. I denna servlet har man möjlighet att anpassa TS2 enligt abonnentens kommandon.

ut i realtidssändningarna som UDP Multicast. PVR-funktionaliteten i Hibox Systems middleware-server är implementerad som Java-servlet och använder Apache Tomcat. Därför är den naturliga utgångspunkten att använda Java för implementationen redan från början.

En enkel lösning på tidsförflyttning skulle vara att när man tar emot ett kommando att spola framåt (eller bakåt) helt enkelt går igenom filen tills nästa (eller föregående) brytpunkt/GOP hittas. Att flytta bakåt understöds egentligen inte av MPEG-standarden eftersom tidsinformationen kommer att minska, men exempelimplementationen har visat mottagaren klarar av det till viss mån. Eftersom detta skall göras över ett nätverk och i realtid på användarens kommando måste man snabbt kunna hitta rätt ställe, och problemet här kan vara att det tar för länge att leta upp nästa GOP. Eftersom nya GOP förekommer relativt sällan i bitströmmen kan det vara bättre att leta efter transportpaket med brytpunktsflaggan satt, istället för att söka igenom enskilda transportströmpaket. Transportpaketets startindikator för nyttolast kan också möjligtvis användas för att enkelt få reda på var en ny GOP, eller nytt fullständigt PES-paket börjar – här krävs dock att en ny GOP verkligen alltid börjar i dessa paket. Denna metod är vad som huvudsakligen används i mediaspelarprogram för datorer – exempelvis xine-lib[6].

Om det blir för beräkningsmässigt krävande att i realtid söka upp rätt position kunde en lösning vara att göra en offline-indexering av transportströmmen, för att få reda på i vilka positioner av filen lämpliga startpunkter hittas. Här är problemet att man ändå måste läsa igenom indexfilen för att hitta en lämplig position, och man måste hålla reda på var man befinner sig i uppspelningen. Indexeringen måste också göras i något skede, frågan är när – direkt efter att inspelningen slutat eller först när en klient börjar spela upp materialet?

[6]http://www.xinehq.de

Snabbspolning är däremot mera utmanande att implementera – man kommer här att vara tvungen att göra förändringar på MPEG-strömmen. En möjlighet kunde vara att endast sända I-rutor, och eventuellt fylla ut med tomma P-rutor mellan för att variera hastigheten på snabbspolningen. Om man t.ex. tar endast I-rutor av sekvensen IBBPBBPBBPBB kommer uppspelningen att gå tolv gånger snabbare än normalt. Genom att sätta in en tom P-ruta och få en sekvens IPIPIPIP... kunde uppspelningen gå sex gånger snabbare. Problemet kan dock vara att uppspelningen inte ser naturlig ut som snabbspolningen på en videobandspelare, samt att funktionen blir beräkningsmässigt krävande. Mest avancerat skulle vara att fullständigt koda om MPEG-strömmen och eventuellt filtrera videon för att få en mjuk snabbspolning.

Den efterfrågade funktionen för att flytta önskat antal sekunder (i praktiken millisekunder) kommer att kräva att man läser in transportströmmens PCR-information. PCR-informationen, som är en momentan sampling av enkoderns klocka på 27 MHz, kan sedan jämföras mellan olika positioner och räknas om till tidsskillnad.

2.2 HTTP-protokollet

2.2.1 Historia och bakgrund

HTTP – HyperText Transfer Protocol – är idag det kanske mest använda protokollet för datakommunikation på Internet, i och med att det används för att överföra data på World Wide Web, ofta i form av HTML med tillhörande bild- och mediafiler som tillsammans utgör en websida. Chris Shiflett berättar i "HTTP Developer's Handbook" [45], ytligt om historien bakom HTTP; det ursprungliga förslaget, "Information Management: A Proposal", presenterades av Tim Berners-Lee i mars 1989. HTTP är således ett relativt nytt protokoll, i förhållande till andra applikationsprotokoll som exempelvis Telnet[7] från 1983 och FTP[8] från 1985. Tim Berners-Lee, som kan anses vara en av de personer som har haft störst betydelse för att göra World Wide Web och Internet till vad det är idag, presenterade förslaget åt CERN (europeiska organisationen för partikelfysikforskning). Shiflett [45] berättar vidare om den första implementationen, World Wide Web-projektet och HTTP 0.9, som utvecklades utgående från detta. Systemet var endast tänkt för att överföra och presentera text, och HTTP 0.9 understödde endast en förfrågan – `GET` – för att hämta en resurs från HTTP-servern.

När intresset för World Wide Web-projektet ökade insåg man snart att HTTP 0.9 var för begränsat och behovet av en utökad implementation resulterade 1996 i HTTP 1.0[9]. Shiflett [45] nämner som de viktigaste tilläggen `POST`-metoden, som möjliggör för klienten att sända data till servern, från t.ex. webformulär. HTTP 1.0 introducerar headers, pakethuvud, som används för att överföra metainformation som berättar bl.a. vilken typ av data som kommer att sändas. Tack vare detta möjliggörs bl.a. bilder inbäddade i websidor genom att klienten kan följa hyperlänkar i HTML-koden och sända förfrågan om en bild. Utöver dessa funktioner tillhandahåller HTTP 1.0 möjligheter för att underlätta mellanlagring (caching) samt grundläggande autentikering. Det var i detta skede World Wide Web började bli verkligt känt bland allmänheten, till stor del som följd av dessa funktioner.

Den senaste versionen är HTTP 1.1[10] från 1999 och det är vad som nästan uteslutande används idag. Shiflett [45] beskriver det tillägg som kanske haft störst betydelse – Host-headern. Denna möjliggör att ansluta flera domännamn mot samma IP-adress, genom att varje Host är associerad med en separat hemkatalog. Tidigare hade den enda möjligheten varit att ha en IP-address för varje domännamn som skall ha en webtjänst. Vidare finns det möjlighet att endast ladda ner en delmängd av innehållet, vilket är användbart bl.a. för strömmande media, samt för att i vissa fall spara på bandbredd.

[7] Teknisk redogörelse för Telnet, http://www.faqs.org/rfcs/rfc854.html
[8] Teknisk redogörelse för FTP, http://www.faqs.org/rfcs/rfc959.html
[9] Teknisk redogörelse för HTTP 1.0, http://www.faqs.org/rfcs/rfc1945.html
[10] Teknisk redogörelse för HTTP 1.1, http://www.faqs.org/rfcs/rfc2616.html

2.2.2 Tekniskt om HTTP

Figur 2.3: Förenklad OSI-modell, Shiflett[46], s. 26

Uppbyggnaden av protokollen behandlas av Shiflett [46] med en förenklad OSI-modell, enligt figur 2.3, i fyra lager istället för de egentliga sju – applikation-, transport-, internet- och nätverkslager. HTTP är högst upp i modellen på applikationsnivån. Tillsammans med HTTP används endast TCP (Transmission Control Protocol) på transportnivån och IP (Internet Protocol) på internetnivån. Standard TCP-port för HTTP är 80. TCP på transportnivån upprätthåller en anslutning, ett "rör" mellan klienten och servern som HTTP-trafiken skickas genom och kommer ut oförändrad på andra sidan. Internetnivån med IP-protokollet delar upp informationen från transportnivån i paket och ser till att de levereras till rätt mottagare. Det är tack vare IP-protokollets routingtabeller som paketen kan levereras snabbt och problemfritt till mottagare var som helst på Internet. Nätverksnivån kan väljas fritt, så länge IP och TCP går att använda – exempelvis Ethernet, DSL eller PPP över telefonlinje, eller trådlöst WiFi.

Slutligen kan nämnas DNS, Domain Name System, som på WWW sköter om att översätta domännamn (inskrivna i webläsare) till en IP-adress som används av IP-protokollet. HTTP är dock inte beroende av DNS.

2.2.2.1 Förfrågningar och respons

HyperText Transfer Protocol är som många andra högnivåprotokoll helt textbaserat och läsligt. Protokoll på lägre nivå är i allmänhet binära, och även om textbaserade protokoll förbrukar större bandbredd kan det anses vara värt med tanke på enkelhet i implementation och felsökning. Man kan t.ex. använda en telnet-klient för att direkt kommunicera med en HTTP-server.

För att identifiera en resurs används av HTTP ett adressformat som kallas URI eller URL, för Uniform Resource Indentifier resp. Uniform Resource Locator. Enligt Shifletts [46] beskrivning kan man ge ett förenklat exempel:

```
http://ourserver.fi/subdir/index2.php?session=5
```

där `http` indikerar protokoll HTTP, `ourserver.fi` är ett domännamn som DNS översätter till serverns IP-adress (IP-adressen som sådan går att använda på samma sätt) och `subdir` är en underkatalog med samma namn. Ett filnamn indikeras av `index2.php` och man överför till HTTP-servern variabeln `session` med värde 5. I allmänhet används denna variabel av filen som hämtas, för att t.ex. påverka innehållet i dynamiska websidor.

Det finns åtta förfrågningar och närmare 50 responskoder. Att beskriva alla i detalj är utanför denna rapports omfång men det centrala följer. Förfrågningarna finns detaljerat beskrivna av Shiflett i kapitel “HTTP Requests” [47] i “HTTP Developer’s Handbook”. I HTTP 1.1 består alla förfrågningar av en förfrågningsrad, en Host-rad samt eventuellt en eller flera förfrågningsspecifika headers, som alla termineras med CRLF (Carriage Return + Line Feed).
De förfrågningar som understöds i HTTP 1.1 och deras funktion är följande:

- `GET` – Den ursprungliga idén bakom HTTP; efterfrågar en specifik resurs/fil utgående från användarinmatad URL, eller bara “/”, som toppkatalogen. Tillsammans med `GET` kan överföras variabelnamn med värden, som kan användas av servern.

- `POST` – Används för att sända data till en HTTP-server i formen variabel=värde, i likhet med `GET`. Skillnaden är dock att `POST` sänder informationen i meddelandet, istället för resursfältet. Fördelen är att variablerna inte är synliga i t.ex. en webläsare, vilket kan ha säkerhetsmässig betydelse.

- `PUT` – Påminner om `POST`, men sänder istället för variabler en resurs som kan lagras som en fil. HTTP-servern skapar om möjligt denna fil, vilket naturligtvis kräver skrivrättigheter och kan utgöra en säkerhetsrisk.

- `DELETE` – Beordrar radering av fil på HTTP-servern. Även här bör säkerhetsaspekter beaktas.

- `HEAD` – Bör resultera i samma HTTP-headers som `GET`, men utan själva innehållet. Detta är användbart vid t.ex. felsökning, och för att undersöka om en resurs existerar på servern utan att ladda ner den.

- `TRACE` – Diagnostikmetod som påminner om IP-protokollets verktyg `traceroute`, vilket visar vilka routrar som paketen slussas genom. `TRACE` visar på HTTP-nivå, vilka proxy-servrar en anslutning går genom.

- `OPTIONS` – Metod för att visa vilka förfrågningar en HTTP-server understöder. Servern kommer att svara med en `Allow`-header som listar förfrågningarna.

- `CONNECT` – Specialmetod som möjliggör för mellanservrar att öppna en transparent anslutning till egentliga servern. Används t.ex. för krypterade anslutningar med SSL och TLS.

Till förfrågan fogas i allmänhet ett antal headers. Shiflett [47] skriver att det existerar 19 förfrågningsheaders. Några förfrågningsheaders som t.ex `Accept` (listar mediatyper) och `Accept-Charset` (listar teckenuppsättningar) används tillsammans med klientens förfrågningar för att beskriva vilka typer av data som kan tas emot. En header som kan vara av betydelse för strömmande media över HTTP är `Range`, med vilken kan specificeras ett antal tecken av innehållet, t.ex. `Range: 0-187` skulle ge de första 188 tecknen vilket är storleken av ett MPEG-transportpaket.
En typisk förfrågan från en klient kan se ut enligt följande:

```
GET / HTTP/1.1
Host: ourserver.fi
User-Agent: Mozilla/5.0 (X11; U; Linux i686; en-US; rv:1.8.0.7) Gecko/20060830
↪ Firefox/1.5.0.7 (Debian-1.5.dfsg+1.5.0.7-2 bpo.1)
Accept: text/xml,application/xml,application/xhtml+xml,text/html;q=0.9,text/plain;
↪ q=0.8,image/png,*/*;q=0.5
Accept-Language: en-us,en;q=0.5
Accept-Encoding: gzip,deflate
Accept-Charset: ISO-8859-15,utf-8;q=0.7,*;q=0.7
Keep-Alive: 300
Connection: keep-alive
```

Servern kommer alltid att besvara förfrågan med en HTTP-respons. Responsen följer samma modell som förfrågningarna, och Shiflett [48] beskriver dem detaljerat i kapitlet “HTTP Responses”. Responsen består dels av en statusrad, dels av diverse headers och slutligen, eventuellt, efterfrågat innehåll. Det finns fem typer av statuskoder som Shiflett [48] förklarar enligt följande:

- Informativa (100-199) – används sällan och innehåller endast två koder, `100 Continue` och `101 Switching Protocols`.
- Framgång (200-299) – indikerar att en förfrågan behandlats utan problem. Ett exempel är `200 OK` som är det normala svaret när en resurs efterfrågats med `GET`. `206 Partial Content` är ett annat exempel, som innehåller den del av materialet som efterfrågats med `Range`-headern.
- Omdirigering (300-399) – möjliggör för HTTP-servern att omdirigera klienten till en annan URL om den har information om vart en resurs flyttats. Exempel är `301 Moved Permanently` och `307 Temporary Redirect`.
- Klientfel (400-499) – indikerar att förfrågan inte kan uppfyllas p.g.a. ett fel från klientens sida. Det kan vara t.ex. `400 Bad Request` som innebär att servern in-

te förstod förfrågan. Det kanske mest kända är `404 Not Found`, som innebär att resursen inte hittades vilket på WWW i allmänhet beror på en felaktig länk.

- Serverfel (500-599) – indikerar att förfrågan inte kan uppfyllas p.g.a. ett fel från serverns sida. `500 Internal Server Error` och `503 Service Unavailable` är några exempel på typiska serverfel.

I likhet med förfrågningar kan responsen även ha en eller flera headers beroende på statuskod. Några exempel är `Location` som används för omdirigering och anger ny URL, `Server`, som berättar vilken mjukvara som används. En typisk respons kan se ut enligt följande:

```
HTTP/1.1 200 OK
Date: Tue, 09 Jan 2007 08:33:26 GMT
Server: Apache/1.3.29 (Unix) PHP/4.4.1 mod_ssl/2.8.16 OpenSSL/0.9.7g
X-Powered-By: PHP/4.4.1
Transfer-Encoding: chunked
Content-Type: text/html

73
<html>
<title>Enkel websida</title>
<body>
Exempel på HTML-kod
</body></html>

0
```

Det finns ett antal övriga header-fält som klassificeras som generella headers och entitets-headers. Shiflett [49] skriver om de generella header-fälten att både förfrågningar och respons kan använda dem. Några användningsområden är styrning av mellanlagring, cache, i klient och eventuella proxy-servrar. Där kan man med `Cache-Control`-headern t.ex. med hjälp av instruktionen `no-store` förhindra att resursen lagras, och med `max-age` ange i sekunder hur länge resursen skall vara giltig. Vardera partens tid och datum kan sändas med `Date`-headern, och anslutningen kan med `Connection`-headern hållas öppen eller avslutas med `Keep-Alive` resp. `Close`. HTTP 1.1 har `Keep-Alive` som standard.

Slutligen beskriver Shiflett [49] entitets-headerfälten, som bl.a. berättar för klienten hur meddelandets innehåll skall behandlas. `Content-Encoding` används för att visa om meddelandet är komprimerat och på vilket sätt – `compress`, `deflate` och `gzip` är giltiga metoder. `Content-Length` indikerar hur många tecken meddelandet innehåller. Vid caching kan `Last-Modified` vara användbar för att kontrollera om en resurs modifierats sedan senaste åtkomst. Slutligen bör nämnas `Content-Type`, som berättar vilken typ

av data meddelandet innehåller – ett normalt HTML-dokument som används på WWW har typen `text/html`, och de flesta filtyper har en standardiserad typ tilldelad av IANA (Internet Assigned Numbers Authority).

2.2.2.2 Konceptet med tillståndslöst (eng. stateless) protokoll

En viktig egenskap Shiflett [50] beskriver är att HTTP-protokollet är tillståndslöst. Det innebär att HTTP-protokollet på inget sätt håller reda på vilka förfrågningar en klient gjort tidigare. Det enda en HTTP-server per definition gör är att behandla de enskilda förfrågningarna. Därför har det blivit nödvändigt att bygga sig runt dessa begränsningar med metoder för att hålla reda på vilka förfrågningar som hör till vilken klient. Man kan givetvis särskilja klienterna enligt IP-adress, men det säger ingenting om t.ex. var i ett web-gränssnitt användaren befinner sig. Dessutom kan många klienter dela samma IP-adress genom NAT (Network Address Translation) – samt vissa proxyservrar[11] kan göra att samma klient får olika IP-adresser mellan förfrågningar.

Shiflett [50] behandlar ett antal populära metoder för sessionshantering. Ett vanligt sätt på WWW är s.k. cookies, en liten textsträng som lagras i webläsaren och kan sändas tillsammans med förfrågningar. Cookies är domänspecifika och kan användas bl.a. för att identifiera en autentikerad användare och lagra individuella inställningar för websystem. Användningen av dessa har debatterats, p.g.a. säkerhetsaspekter samt integritet och kartläggande av användarnas beteende.

Utöver detta är det möjligt att använda GET- och POST-förfrågningarna för sessionshantering. Genom att använda ett skriptspråk som t.ex. PHP kan man generera en unik sträng för att hålla reda på sessionerna, och skicka över informationen som en URL-variabel i GET-förfrågan t.ex. enligt följande (baserat på Shifletts [50] exempel):

```
GET /index2.php?session=5 HTTP/1.1
Host: ourserver.fi
```

I detta fall kunde man tänka sig att `index.php` innehållit en funktion som håller reda på aktuella sessioner, och tilldelar denna klient session-variabel 5. På samma sätt är det möjligt att sända informationen med POST-förfrågan, men då utan att den syns t.ex. i webläsarens adressfält. För att sammanfatta Shifletts [50] genomgång av tillståndshantering kan sägas att URL-variabler är den metod som lämpar sig bäst för att identifiera enskilda sessioner, medan cookies som lagras för längre tid (enligt önskemål) i klienten är användbara t.ex. för att spara inställningar. POST-variabler används i allmänhet bara vid inmatning i webformulär.

[11] Server som fungerar som mellanhand och kan utföra t.ex. caching och innehållsfiltrering

2.2.2.3 Implementationer av HTTP

Det existerar en uppsjö implementationer av servermjukvara för HTTP. Dessa kan vara generella webservrar eller specialiserade för något användningsområde. Wikipedia [51] listar ett stort antal i jämförelse. Skillnader kan vara t.ex. licens – fri (t.ex. Apache, GPL eller BSD) eller proprietär, operativsystemstöd samt funktioner för säkerhet och dynamiskt innehåll.

Den i särklass mest kända universal-webservern är Apache HTTP Server[12]. På projektets hemsida [52] kan man läsa att projektet kom till i början av 1995 med kodbas från NCSA:s (National Center for Supercomputing Applications) fria HTTP-server, som vid det tillfället saknade underhåll. Apache HTTP Server är idag den mest använda webservern på Internet, och finns till de flesta operativsystem. Den anses även vara mycket säker och pålitlig. Operativsystemet Linux tillsammans med Apache HTTP Server, den fria databasen MySQL och skriptspråket PHP (PHP: Hypertext Preprocessor) kallas LAMP-plattformen och har blivit något av en vedertagen modell för webserversystem som helhet.

Microsofts lösning, IIS (Internet Information Services)[13] är en integrerad del av Windows-operativsystemfamiljen. Enligt en artikel på Wikipedia [53] fanns den första versionen som gratis tillägg till NT 3.51. Idag är man uppe i version 7.0 som kommer att levereras med Windows Vista. IIS, som förutom HTTP-server även består av FTP-, SMTP- och NNTP-server, kan använda Microsofts skriptspråk ASP (Active Server Pages) och är nära integrerad med .Net-miljön. IIS finns endast för Windowsmiljö och har enligt artikeln på Wikipedia [53] varit behäftad med ett antal säkerhetshål.

Java-servlets och Apache Tomcat: Allt eftersom WWW utvecklades från statiska sidor med text, bilder och media till s.k. web-applikationer som skall kunna dynamiskt anpassa innehållet uppstod ett behov av att kunna kontrollera hur detta innehåll genereras. Även om man ofta kan åstadkomma mycket dynamiska web-applikationer med serverbaserade skriptspråk som PHP och ASP ger de inte alltid tillräcklig kontroll.

Java-servlets ger tillgång till större kontroll samt bättre tillståndshantering. Wikipedia-artikeln [54] om servlets förtäljer att ett första förslag på lösning för mera dynamisk HTTP introducerades av Sun Microsystems i juni 1997. Detta har utvecklats till den servlet-specifikation som används idag. Med hjälp av en servlet får man full tillgång till HTTP-förfrågan med hjälp av ett `ServletRequest`-objekt, och skapar ett `ServletResponse`-objekt enligt applikationens behov. För dessa finns abstrakta metoder som man implementerar – t.ex. `doGet` som körs vid `GET`-förfrågan. JSP (JavaServer Pages) är en annan teknologi som bygger på servlets, med ett skriptspråk som kompileras till servlets.

Java-servlets kan inte presenteras med en vanlig webserver – en servlet utgör som

[12] http://httpd.apache.org
[13] http://www.microsoft.com/iis

sådan en webserver. Istället används en s.k. webcontainer som tillhandahåller den nödvändiga infrastrukturen för att en servlet skall kunna fungera, och den kanske mest kända av dessa är den fria Apache Tomcat[14]. En artikel på Wikipedia [55] om Tomcat nämner att Tomcat ursprungligen var referensimplementation för servlet-container från Sun Microsystems. Denna kodbas donerades sedan till Apache-stiftelsen. Idag stöder Apache Tomcat förutom servlets också JSP via kompilatorn Tomcat Jasper. Tomcat finns tillgänglig för de flesta operativsystem och kan även fungera som normal webserver.

2.2.3 HTTP och strömmande media, Hibox Systems lösning

Strömmande media över HTTP fungerar i praktiken likadant som allt annat innehåll – TCP ser till att innehållet överförs felfritt och i rätt ordning. Denna fördel kan dock tänkas vara till nackdel om överföringskanalen är opålitlig och paket tappas bort. Då kommer TCP att sända den borttappade informationen igen, medan mottagaren blockeras av sitt transportlager tills informationen tagits emot, vilket kan resultera i att uppspelningen avbryts. Med hjälp av lämpligt stort buffertminne i mottagarens applikationslager kan dock detta problem undvikas. Man brukar istället för HTTP över TCP ofta använda UDP (User Datagram Protocol) för strömmande media. Även om detta enklare protokoll saknar inbyggd fel- och leveranskontroll är ett enstaka borttappat paket sällan ett problem i fallet med strömmande media. UDP, till skillnad från TCP, saknar inbyggd flödeskontroll vilket inte är nödvändigt vid realtidssändningar. När man däremot överför förinspelat videomaterial som spelas upp i realtid måste mottagaren kunna styra hur snabbt materialet skall överföras för att undvika överfyllda buffrar. I detta fall är TCP lämpligt eftersom det tillhandahåller flödeskontroll som kan anpassa dataströmmen till mottagarens behov.

Något som i vissa fall kan tänkas orsaka problem är TCP-protokollets slow-start-metod som används för att utreda nätverkets kapacitet och undvika nätverksstockning. Vid start av överföring ökas paketstorleken gradvis tills paketförlust inträffar. På samma sätt anpassas paketstorleken kontinuerligt för att utnyttja kapaciteten maximalt. Med många pågående överföringar kan detta ställa till problem när en ny överföring påbörjas, eftersom det kan ta en tid för alla TCP-anslutningar att minska paketstorleken och sända eventuellt förlorad data pånytt.

Hibox Systems har implementerat PVR-funktionalitetens uppspelning av videomaterial över HTTP. Detta underlättas av att middlewaresystemets spelarfunktionalitet är en servlet som ger full kontroll över hur förfrågningarna behandlas och vad som sänds tillbaka till klienten. För att införa tillståndshantering används GET-variabler, i form av ett unikt `session`-nummer som vardera klient tilldelas av middleware-systemet. Detta nummer sänds som GET-variabel tillsammans med förfrågningarna. Som GET-variabel sänds

[14]http://tomcat.apache.org

även ett `recording`-nummer som servlet-programmet associerar med rätt fil. Ett exempel på URL kan vara:

```
http://ourserver.fi/player/Player?session=2&recording=1357
```

Utifrån detta kommer servlet-programmet att leverera inspelning med ID 1357, och dessutom hålla reda på att klienten med session 2 håller på att spela upp den.

Genom att lägga till variabeln `action` styrs uppspelningen. Denna variabel kan ursprungligen anta värdena `skip`, `seek` och `pause-resume`. `skip` kräver dessutom en heltalsvariabel `bytes`, som kan vara positiv eller negativ beroende på om man skall förflytta sig `bytes` tecken framåt eller bakåt, och på samma sätt kräver `seek` en variabel `position` som får uppspelningen att fortsätta från en absolut position. Ett exempel på användning:

```
http://ourserver.fi/player/Player?session=2&recording=1357
↪ &action=skip&bytes=-20000
```

Kommer att få uppspelningen att hoppa 20000 tecken bakåt i transportströmmen, samt

```
http://ourserver.fi/player/Player?session=2&recording=1357
↪ &action=seek&position=100000
```

Fortsätter uppspelningen från tecken 100000. Inläsningen av transportströmfilen sker med gränssnittet `java.io.RandomAccessFile`, vilket möjliggör åtkomst till godtycklig position. I detta skede kommer det att bli nödvändigt att hitta rätt ställe i transportströmmen att fortsätta uppspelningen, eftersom det ursprungliga systemet fortsätter direkt oberoende av vilka datastrukturer som råkar finnas i den positionen. Det är detta som gör att MPEG-avkodaren i mottagaren får problem att tolka strömmen, med resulterande blockfel.

2.3 Jämförelse av strömmande media över HTTP mot RTP-protokollfamiljen

RTP (Real-time Transport Protocol) utgör grundpelaren i en avancerad uppsättning applikationsprotokoll som ofta används för olika former av strömmande media över nätverk och Internet. Colin Perkins tar upp bakgrunden till RTP i "RTP: Audio and Video for the Internet" [56]. Den ursprungliga drivkraften att utveckla ett sådant protokoll var intresset för audio- och videokonferenser. Så tidigt som 1977 presenterades NVP (Network Voice Protocol), vilket kan anses vara en direkt föregångare till RTP. På samma sätt som RTP används över det anslutningslösa UDP-protokollet, användes en "okontrollerad paketöverföring" på ARPANET för datakommunikationen. Ett allt mera utbrett Internet och högre hårdvaruprestanda ledde i början av 1990-talet till större möjligheter och behov för strömmande media. Detta resulterade i RTP-protokollet, som utvecklades av IETF (Internet Engineering Task Force) mellan 1992 och 1996 och publicerades i januari 2006 som RFC 1889[15]. Denna standard ersattes senare i juli 2003 av den förbättrade RFC 3550[16].

2.3.1 Dataöverföring med RTP

Perkins [56] beskriver vidare principen bakom RTP; att tillhandahålla en komplett protokollstruktur för realtidsöverföring av media. Utöver paketering av media erbjuder RTP bl.a. synkronisering av strömmar, feldetektering och korrigering samt återkoppling och rapportering av överföringskvalitet. RTP har mekanismer för att hålla reda på deltagare i en session, något som det tillståndslösa HTTP-protokollet i grunden saknar. Eftersom RTP ofta används tillsammans med IP multicast är det nödvändigt att veta vilka klienter som skall höra till mottagargruppen. IP multicast möjliggör att sända samma data till ett mycket stort antal användare utan att ta upp större bandbredd än vid sändning till endast en.

Några viktiga principer för RTP beskrivs av Perkins [57] i kapitlet "The Real-time Transport Protocol". Robusthet och tidskorrekt leverans eftersträvas. Detta kan åstadkommas genom att protokollet som sådant sköter om hur informationen paketeras och levereras över det primitiva UDP-protokollet istället för att förlita sig på ett avancerat transportprotokoll som TCP. RTP-protokollet tar på sig ansvaret för korrekt leverans, i den mån detta är kritiskt. Olika profiler kan definieras beroende på användningsområden och behov – exempelvis audio- och videokonferens jämfört med ett VOD-system.

[15] http://tools.ietf.org/html/rfc1889
[16] http://tools.ietf.org/html/rfc3550

2.3.2 RTSP styr uppspelningen

Vidare uppmärksammar Perkins [56] att RTP endast är ett protokoll för överföring av media i realtid – för att bl.a. starta upp en RTP-session behövs ett annat kommunikationsprotokoll. För videokonferenssystem och IP-telefoni är det dominerande protokollet SIP (Session Initiation Protocol), medan multimedia-applikationer har ett eget specialiserat protokoll – RTSP. RTSP (Real-Time Streaming Protocol) publicerades av IETF 1998 som RFC 2326[17], och möjliggör kontroll av mediaströmmen med motsvarande uppsättning funktioner som en videobandspelare. De VOD-system som används för IPTV använder i allmänhet RTSP, vilket också set-top-boxarna understöder. Protokollet är enkelt att använda från klientens sida och har mycket gemensamt med HTTP. I likhet med HTTP är RTSP textbaserat och använder snarlik syntax med förfrågningar och respons. Specifikationen från IETF [58] beskriver elva förfrågningsmetoder:

```
DESCRIBE, ANNOUNCE, GET_PARAMETER, OPTIONS, PAUSE, PLAY, RECORD, REDIRECT,
SETUP, SET_PARAMETER, TEARDOWN
```

Responsen från servern kommer att innehålla en statuskod som i stort sett är samma som används i HTTP, t.ex. `200 OK` eller `404 Not Found`. En guide till RTSP på myIPTV.org [59] visar följande exempel för att starta uppspelningen börjande med en `SETUP`-förfrågan från klienten:

```
SETUP rtsp:<filename> RTSP/1.0
CSeq: 1
Transport: MP2T/H221/UDP;unicast;destination=000!192.168.0.1!192.168.0.2!1234
x-historyLog:
x-mayNotify:
```

Därefter kommer RTSP-servern att svara med en statuskod och tilldela sessionen ett identifikationsnummer:

```
RTSP/1.0 200 OK
CSeq: 1
Session: 123456789101112131415
Transport: MP2T/H221/UDP;unicast;destination=000!192.168.0.1!192.168.0.2!1234
```

Klienten kan nu starta uppspelningen av sessionen; `Range`-parametern meddelar mellan vilka tider i innehållet uppspelningen skall ske – i detta fall från början till slut. `Scale` ger skalningsfaktor för uppspelningshastighet – ett negativt värde ger uppspelning baklänges. `CSeq`-parametern är ett löpnummer per förfrågan som kommer att vara samma i responsen:

[17] http://tools.ietf.org/html/rfc2326

```
PLAY rtsp:<filename> RTSP/1.0
CSeq: 3
Range: npt=0-
Scale: 1
Session: 12345678910111213141 5
```

RTSP är tillsammans med RTP ett serverintensivt system eftersom många av dessa operationer kräver förändringar i mediaströmmen – exempelvis för att spela en videofil baklänges. I förhållande till RTP är RTSP ett fristående applikationssprotokoll och kan enligt specifikationen [58] användas över antingen TCP eller UDP.

2.3.3 RTCP för kvalitetskontroll

En integrerad del av RTP är RTCP (RTP Control Protocol) som används för att föra statistik över bl.a. kvaliteten på dataöverföringen. Perkins behandlar ingående systemet i kapitlet “RTP Control Protocol” [60]. RTCP använder UDP i likhet med RTP, på efterföljande udda port (RTP använder jämna portnummer). Fem pakettyper finns definierade:

- Mottagarrapport (Receiver Report) – bl.a. antal förlorade paket och nätverksfördröjning. Detta är en viktig del av RTCP och kan bl.a. användas för att anpassa bithastigheten på sändningen i förhållande till nätverkets kapacitet och stabilitet.
- Sändarrapport (Sender Report) – innehåller tidsinformation för synkronisering av audio- och videoströmmar. Behövs eftersom man i allmänhet har skilda audio- och videoströmmar, samt i vissa fall flera audioströmmar för mottagaren att välja på.
- Källbeskrivning (Source Description) – information om en deltagare att lagra i serverns deltagardatabas. Denna information fylls i av användaren och kan bestå av bl.a. e-mail-adress, telefonnummer och hemadress.
- Medlemskapshantering (Membership Management) – BYE-paket som indikerar att en deltagare lämnar sessionen.
- Applikationsspecifika funktioner.

2.3.4 I jämförelse med HTTP

RTP, som är ett komplext och specialiserat protokoll, har både likheter och olikheter med det väldigt generella HTTP-protokollet. HTTP saknar motsvarighet till RTCP, och en del av den funktionaliteten tillhandahålls istället av det underliggande TCP-protokollet. TCP sänder t.ex. förlorade paket igen, oberoende om det är önskat eller ej. En sådan funktion måste med RTP göras som applikationsspecifikt tillägg i RTCP.

RTSP påminner mycket om HTTP men HTTP saknar de flesta av de bakomliggande funktionerna – t.ex. skalningsfaktorn på uppspelningen. Med hjälp av servlet-teknologin går det dock att skapa funktioner som motsvarar RTSP. Hibox Systems Player-servlet har redan funktioner för att flytta framåt/bakåt i transportströmmen samt pausa uppspelningen. Målet i detta diplomarbete är att sätta till den bakomliggande funktionalitet som får uppspelningen att starta på rätt position så att mottagaren inte blir förvillad, samt att sätta till funktioner för att flytta uppspelningen önskad tid (millisekunder) framåt/bakåt precis som med RTSP.

Kapitel 3

Implementation av system för förflyttning

3.1 Serversida

3.1.1 Första undersökningar av transportströmmen

De första verktygen som utvecklades gjordes för att kunna analysera och förstå transportströmmen. Exempel på strömmar erhölls från Hibox Systems. Dessa var dels från PVR-systemet (delmängder av fullständiga transportströmmarna), dels fullständiga strömmar tagna direkt från digital-TV-sändningarna. Eftersom Hibox Systems servermjukvara är utvecklad i Java ansågs detta vara ändamålsenligt för hela programutvecklingen. Centralt för alla de grundläggande undersökningsverktygen är att transportströmfilen öppnas som en `java.io.RandomAccessFile`, ett programmeringsgränssnitt som möjliggör binär läsning och skrivning av en fil och tillåter att flytta läsningen/skrivningen till godtycklig position.

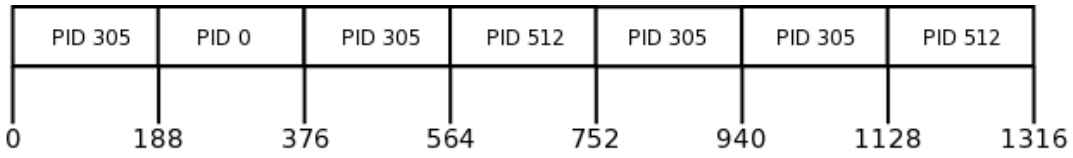
Det allra första verktyget (benämnt `readts`) går igenom transportströmmen paket för paket och skriver ut information om dem i formen:

```
TS packet with PID 305 (0x131) found at pos: 0x8a914, diff: 188
TX error: 0, payload start: 0, TX prio: 0
CA: 0, adaptation field: 1, continuity count: 14
```

När programmet hittar synkroniseringstecknet för transportström (0x47) instantieras ett objekt av klass `TSheader` (klassens konstruktor `TSheader(RandomAccessFile TS)`) som tar transportströmmen som argument i form av ett `RandomAccessFile`-objekt, och ger lättåtkomlig information om pakethuvudet. Utgående från utskriften kan man se hur de enskilda paketen är distribuerade och hur paketens bitfält är satta. Positionen i filen skrivs ut för att man enkelt skall kunna hitta transportpaketet om man öppnar filen i en hexeditor.

Hexeditor har varit till stor hjälp för att förstå transportpaketens format, samt formatet för den paketerade elementära strömmen.

Därefter gjordes ett verktyg (benämnt `listPIDs`) som listar alla PID som hittas i en transportström, tillsammans med förekomst samt procentuell förekomst. Detta verktyg visade sig vara mycket användbart vid vidare analys av transportströmmen. Den procentuella förekomsten gör det enkelt att se vilken PID videoströmmens transportpaket har – dessa utgör mellan 70 och 90 procent av en SPTS (Single Program Transport Stream). Audioströmmen utgör ca 7–10 procent. Figur 3.1 visar transportströmmens uppbyggnad av paket om 188 tecken som identifieras utgående från sitt PID-nummer.

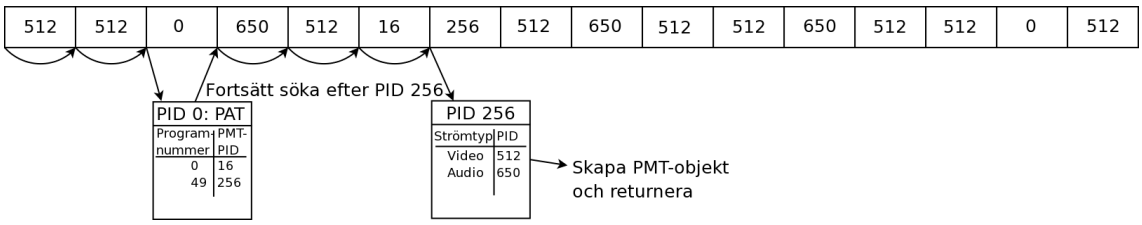


Figur 3.1: Exempel på transportströmmens uppbyggnad

För att bättre förstå den paketerade elementära strömmen gjordes verktyget `dumpvideoPES`, som skriver ut transportpaketets nyttolast från paket med en angiven PID. Programmet traverserar transportströmmen paket för paket, instantierar `TSheader`-objekt och hoppar till nästa om transportströmpaketet inte har önskad PID. Om paketet har rätt PID extraheras nyttolasten av funktionen `static byte [] getPES (TSheader header, RandomAccessFile TS)`. Först måste dock paketet undersökas närmare – om pakethuvudet indikerar att paketet innehåller ett anpassningsfält (adaptation field) skall fältet lämnas bort, annars returneras hela nyttolasten som en binär array. Därefter skrivs denna array ut till en fil, och i fallet att man tagit ut video- eller audioströmmen går denna fil att spela upp i t.ex. VLC.

Det skulle också komma att vara nödvändigt att läsa in PSI (Program Specific Information), dvs. informationen om vilka program transportströmmen består av, samt deras elementära strömmars PID. Eftersom transportströmmarna i detta fall innehåller bara ett program behöver man endast ta reda på PID för dess videoström. Detta implementerades i två klasser – `PAT` och `PMT`, med konstruktorerna `PAT(RandomAccessFile TS)` och `PMT(RandomAccessFile TS)` respektive. Dessa klasser kräver således att `RandomAccessFile`-objektet givet som argument står på rätt ställe i transportströmfilen, början av nyttolasten (vilket det också gör efter att transportström-headern lästs in). Funktionen `static PMT findVideoPMT (RandomAccessFile TS)` implementerades i `dumpvideoPES` för att med ett enda anrop få ut videoströmmens PMT-tabell. Figur 3.2 illustrerar hur funktionen söker upp rätt tabell. Det första funktionen gör är att söka genom transportströmmen efter ett transportpaket med PID 0, dvs. PAT. När ett sådant hittats instantieras ett objekt av klass `PAT`, som i likhet med `TSheader` ger utförlig information

representerad på ett åtkomligt sätt. Om pakethuvudets table_id är noll kan man vara säker på att PAT hittats, och då fortsätter objektet att skapa sina tabeller över tillgängliga program i transportströmmen.



Figur 3.2: Funktionen `findVideoPMT` returnerar första programmets PMT-tabell för en SPTS

Efter att PAT lästs in, antas i detta fall att transportströmmen innehåller endast ett program. PAT innehåller två poster; den ena är program noll som betyder PID för NIT (Network Information Table), vilket inte är relevant i sammanhanget. Därför söker man upp PMT för det program som har programnummer olika noll. När TS-paket med eftersökt PID hittats instantieras ett objekt av klass `PMT`, som läser in programmets information ifall headern konstateras ha table_id av två (indikerar PMT).

PMT-objektet kommer att innehålla två heltalsarrayer av PID respektive strömtyp, och videoströmmen kommer enligt ISO/IEC 13818-1 [42] att ha strömtyp två eller tre. I PMT-objektet lagras heltalsflaggan complete, som berättar att hela den förväntade tabellen lästs in. Tills den satts till ett söker man upp nästa transportströmpaket med programmets PMT-PID, och sätter till det i PMT-tabellen med `PMT`-klassens funktion `void add(RandomAccessFile TS)`. Slutligen måste man söka igenom objektets array av strömtyper efter önskad typ (två eller tre för videoström) och ta PID för denna ström ur motsvarande index i arrayen av PID-nummer.

Denna metod är dock endast användbar på transportströmmar med ett program, om flera existerar kommer den att returnera PMT till det första programmet. Om flera program existerar är man tvungen att ha ett sätt att välja önskat program.

För att undersöka transportströmmens system för tidsreferenser, PCR, samt brytpunkter (splicing point) gjordes verktyget `showPCR`. Programmet börjar med att skapa ett objekt av klass `PMT` för transportströmmens relevanta program. Ur detta objekt kan läsas vilken PID som innehåller PCR-informationen. Programmet går igenom transportströmmen paket för paket och om ett paket har PCR-informationens PID och anpassningsfält skapas ett objekt av klass `PCR`. `PCR`-klassens konstruktor, `PCR (byte [] data)`, tar som argument en teckenarray innehållande anpassningsfältet. Om anpassningsfältets PCR-flagga är satt, fortsätter klassen med att läsa in PCR-informationen. Som ISO/IEC13818-1 [40] beskriver, består PCR-informationen av två delar; PCR_base och PCR_extension, på 33

respektive 9 bitar. Fullständiga PCR-värdet kan utgående från dessa beräknas enligt ekvation 3.1.

$$PCR = PCR\_base * 300 + PCR\_extension \tag{3.1}$$

När de båda fälten för PCR-informationen skulle läsas in upptäcktes en begränsning i Javas hantering av tecken. Java har ingen teckenlös (unsigned) datatyp, vilket gör att tecknen tolkas i intervallet [-128, 127] istället för det önskade [0, 255]. Detta löses genom bitvis AND med 0xFF på de enskilda tecknen. Därefter listas den tillgängliga PCR-informationen i formatet:

```
PCR info found at pos d7e8
Parsing adaptation field size 183
Discontinuity: 0
Random access: 0
ES Prio: 0
PCR: 1
OPCR: 0
Splicing point: 0
Private data: 0
Adaptation field extension: 0
PCR base field: 5213676333
PCR extension field: 108
Full PCR: 1564102900008

Difference in PCR: 1017144 ticks
In mseconds: 37
```

Skillnaden mellan två PCR-fält visas både i klocktickningar och millisekunder. Skillnaden i millisekunder fås enkelt genom ekvation 3.2.

$$\Delta_{t[ms]} = \frac{PCR_2 - PCR_1}{27000} \tag{3.2}$$

Slutligen för att närmare undersöka strömmen på PES-nivå gjordes verktyget `analyzePES`. Detta program använder samma metod som den senare implementerade funktionen `validStartPES`, illustrerad i figur 3.6, för att söka upp PES-nivåns startkodsprefix. När ett prefix hittats undersöks det efterföljande tecknet och utgående från det instantieras objekt som läser in informationen. De nya klasser som skapats är `streamheader`, `sequenceheader`, `GOP` och `pictureheader`, för videoström, sekvens, GOP och bildrutor respektive. Gemensamt för dessa är att deras konstruktorer tar ett `RandomAccessFile`-objekt som argument. Objektets filpekare skall stå på rätt position, efter startprefix och startkod. Vart och ett av header-objekten läser då in datastrukturerna och skriver ut all relevant information, enligt exemplet nedan:

```
Prefix found at pos: de1c: ea, identified as video stream header
PES packet length: 0
Scrambling: 0
Priority: 1
Alignment: 1
Copyright:1
Original/copy: 1
PTS/DTS: 3
ESCR flag: 0
ES rate: 0
DSM trick mode: 0
Additional copy info: 0
PES_CRC: 0
PES extension: 0
PES header length: 10
PTS: 5213798416
DTS: 5213787616

Prefix found at pos: de2f: b3, identified as sequence header
Horizontal size value: 720
Vertical size value: 576
Aspect ratio information: 2
Frame rate: 25
Bit rate: 25003
VBV Buffer size: 568

Prefix found at pos: de7b: b5, identified as extension header
Prefix found at pos: de85: b2
Prefix found at pos: de8f: b8, identified as GOP header
GOP with time 7:17:36
Frame 2, closed GOP: 0, broken: 0, next_start: 0

Prefix found at pos: de97: 0, identified as picture header
Temporal reference: 2
Frame type: I
VBV Delay: 65535
```

3.1.2 Observationer från undersökningarna

Efter att ha kunnat extrahera en spelbar video-PES kunde transportströmmens egenskaper undersökas närmare. I ett tidigt skede konstaterades att DVB-sändningarna aldrig innehåller brytpunkter (splicing points) som en flagga i transportpaketets anpassningsfält – vilket är förståeligt eftersom de i grunden endast är gjorda för att presenteras i realtid och inte för att redigeras. Något som däremot snabbt visade sig ha betydelse var transportpaketets

startindikator för nyttolast, payload_start. Genom att manipulera `dumpvideoPES` att endast skriva ut nyttolast från TS-paket som indikerar start av nyttolast, samt från paket som inte indikerar start av nyttolast kunde följande relevanta observation göras: startkoderna för sekvens och GOP (0x000001B3 resp. 0x000001B8) från den elementära videoströmmen existerar endast i TS-paket med payload_start-flaggan satt. Däremot börjar inte ny GOP i alla TS-paket som har payload_start, men detta är ändå relevant för att kunna hitta lämpligt ställe att fortsätta uppspelningen vid förflyttning.

Om man söker genom transportströmmen efter ett video-PID-paket som har payload_start och extrahera den paketerade elementära videoströmmen därifrån fås en start av videon utan blockfel – i synnerhet om en ny GOP börjar i det paketet. När en ny GOP förekommer föregås den av sekvens-startkoden för ES (0x000001E0-0x000001EF), vilken föregås av PES-ID för videoströmmen. Dessa paket förekommer relativt ofta i transportströmmen och är därmed en synnerligen användbar egenskap för att flytta uppspelningen.

Alla TS-paket med payload_start börjar med PES-ID för videoströmmen, men för att ett TS-paket skall anses lämpligt att starta från bör den efterföljas av sekvens- samt GOP-startkod. För att kunna vara säker på att transportströmpaketet är en giltig startposition utökades `dumpvideoPES`, med funktionen `static boolean validStartPES (byte [] data)`. Funktionen `validStartPES` tar som argument den binära arrayen data, som innehåller nyttolasten från transportpaketet i fråga. Funktionen går igenom arrayen, och upprätthåller en tillståndsmaskin över funna relevanta startkoder och ström-ID.

Transportströmpaket med krypteringsindikatorn satt visade sig däremot sakna dessa egenskaper. Dessa TS-paket (strömmen krypterad med Conax-systemet) innehåller aldrig payload_start, och nyttolasten innehåller heller aldrig startkodsprefixet 0x000001. Detta innebär att det inte är möjligt att hitta lämpliga startpositioner i dessa strömmar utan att avkoda dem.

PCR-informationens upplösning ligger mellan 25 och 40 millisekunder beroende på ström. Den är dock konstant i enskilda strömmar, med variationer på några millisekunder. På PES-nivå förekommer PTS och DTS varierande ofta beroende på ström. Några av exempelströmmarna har videoström-header före varje ny ruta, medan andra har det endast före ny GOP. Före GOP förekommer dock alltid båda PTS- och DTS-fält, vilket enligt ISO/IEC 13818-1 [44] är ett krav för icke-sömlös hopfogning. I det fall att videoström-header föregår varje bildruta, har P-rutor också både PTS- och DTS-fält, vilket är förståeligt eftersom de behöver avkodas före mellanliggande B-rutor.

3.1.2.1 Möjliga metoder för störningsfri flyttning av uppspelningsposition

Utgående från dessa observationer kunde tre olika förslag på lösning av problemet med förflyttning i videoströmmen presenteras:

Förslag 1 – transportströmnivå: En enkel lösning kunde vara att synkronisera transportströmmen så att man börjar sända ut från ett fullständigt transportpaket i närheten av den nya positionen i transportströmmen. Viktigt är också att man sänder det senaste transportpaketet i sin helhet och inte avbryter mitt i det. Meningen med detta är att undvika att mottagaren blir förvillad av felaktiga och oväntade datastrukturer, men kan inte antas påverka mängden blockfel till märkbar nivå.

Förslag 2 – paketerade elementära strömmens nivå: En bättre lösning som inte är mycket mera krävande beräkningsmässigt, är att söka upp ett TS-paket tillhörande videoströmmen som har payload_start-flaggan satt. Här kommer blockfelen att signifikant minska men kan fortfarande finnas eftersom ström-sekvens-GOP-kombinationen av ström-ID och startkoder inte nödvändigtvis i alla dessa paket.

Förslag 3 – elementära strömmens nivå: Denna lösning bygger vidare på förslag 2; först söka upp paket som har payload_start-flaggan satt men vidare kontrollera innehållet för att se om det innehåller en GOP-header. Enligt observationer från exempelströmmarna börjar dessa paket med en PES-ID för videoström, därefter sekvens- och GOP-startkod för ES. Hur ofta dessa paket förekommer varierar beroende på videoström; i några av exempelströmmarna är alla TS-paket med payload_start sådana, medan det i andra förekommer att payload_start-paket börjar även direkt med P-rutor. Genom att se till att starta på ett TS-paket som innehåller sekvens- och GOP-startkod bör uppspelningen fortsätta felfritt.

Om detta inte visar sig fungera tillfredsställande kan det vara nödvändigt att se till att den pågående GOP sänds i sin helhet, innan man fortsätter uppspelningen på ny position. Detta kommer att innebära mera krävande funktioner och möjligen en offline-indexering för att veta var varje GOP börjar, men är i det närmaste garanterat att ge en perfekt övergång till den nya positionen.

En annan sak som kan visa sig orsaka problem är hur mottagaren tolkar de felaktigheter som uppstår i transportströmmens klockreferens (PCR), i transportströmmens kontinuitetsräknare samt i den paketerade elementära strömmens PTS- och DTS-information. I värsta fall finns det risk att den ignorerar paket som har oväntad tidsinformation eller visar fryst bild som ISO/IEC 13818-1 [44] nämner. Dock skulle en sömlös hopfogning kräva mera ingående modifiering av transportströmpaketen.

3.1.2.2 Föreslagen metod för tidsrefererad förflyttning

Man kan utgå ifrån de befintliga skip- och seek-metoderna men sätta till funktionalitet för att söka genom transportströmmen tills beräknad PCR-information hittats. Samma `PCR`-klass som använts i `showPCR` kan återanvändas. Det första steget är att leta upp närmaste PCR-information, och därifrån räkna ut vad som är önskad ny PCR.

Nya GET-parametrar utöver skip och seek kommer att bli nödvändiga, såväl som en parameter för att ange önskat antal millisekunder. Även i set-top-boxen kommer detta att kräva ändringar, både för inläsningen av önskad tid och för att kunna kalla på serversidans funktioner.

3.1.3 Försöksimplementation på Hibox Systems PVR-server

För att kunna utveckla funktionerna erhölls den grundläggande Player-servlet som idag används på systemet. Även om den är del av en större helhet kunde den enkelt modifieras för att fungera fristående.

Något som tidigt visade sig orsaka problem var parallellismen i Player-servleten tillsammans med `java.io.RandomAccessFile`:s sekventiella natur; om Player-servleten läser data att sända till klienten och förflyttning/synkronisering utförs samtidigt kommer den nya positionen att hamna efter korrekt synkroniserad position – med korrupt transportström som följd. Därför är det nödvändigt att stoppa läsningen före man flyttar position och söker upp rätt datastrukturer.

För att kunna testa implementationerna lokalt sattes Apache Tomcat 5.5 tillsammans med Hibox Systems Player-servlet upp fristående på en experimentdator. Det är därefter möjligt att ansluta till servleten med ett mediaspelarprogram, och för det ändamålet används VideoLAN Client, VLC. För att kunna styra uppspelningen skickas GET-variablerna genom adressfältet i en webläsare, i formatet

```
http://localhost/player/Player?session=1&recording=37996
↪ &action=skip&bytes=500000
```

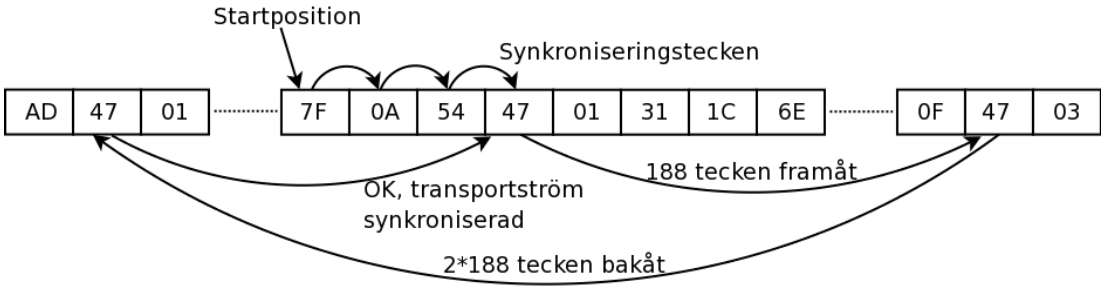
Detta är en följd av att HTTP-protokollet är tillståndslöst, och det finns ingenting som håller reda på varifrån en förfrågan kommit. Servleten håller internt reda på sessioner och tillämpar de mottagna kommandona på önskad videoström.

Utöver detta har implementationerna med jämna mellanrum testats med Kreatels IPTV-mottagare mot Hibox Systems experimentserver. Detta ger möjlighet att se skillnaderna i hur set-top-boxens MPEG-avkodare tolkar avbrott i videoströmmen jämfört med VLC. Kreatels set-top-box visar loggmeddelanden på en nätverksport som man kan ansluta till med t.ex. en telnetklient.

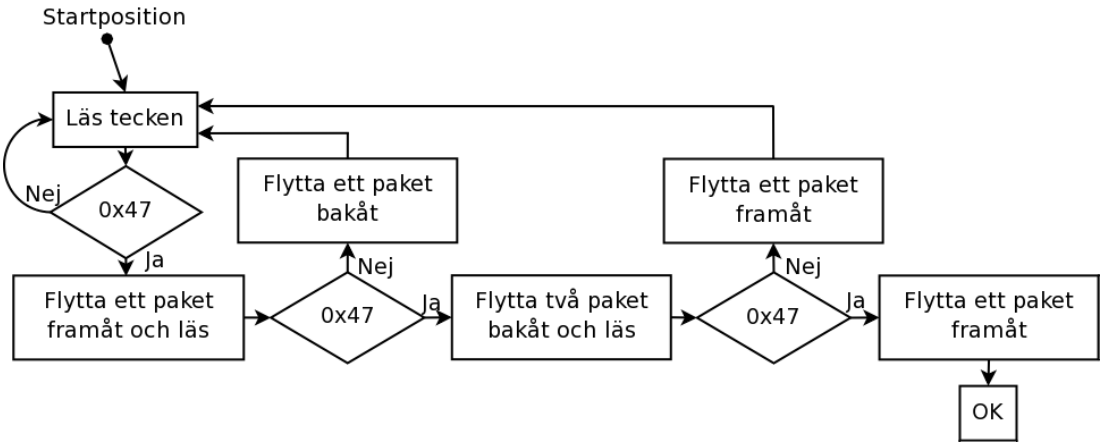
Resultaten är i detta fall väldigt subjektiva; det går inte på enkelt sätt att tekniskt mäta dessa blockfel, även om man tydligt bör kunna se om en metod innebär en förbättring. Något som också är relevant är hur en metod påverkar Kreatel-mottagarens tendens att stanna upp eller helt krascha.

3.1.3.1 Förslag 1 för störningsfri flyttning – transportströmnivå

Synkronisering på transportströmnivå innebär att söka upp början på nästa TS-paket, d.v.s. ett synkroniseringstecken, 0x47, och fortsätta uppspelningen. Dock är problemet att detta tecken inte förekommer endast som synkronisering utan även i nyttolasten. Lösningen är att kontrollera om synkroniseringstecknet förekommer också 188 tecken framåt samt 188 tecken bakåt, vilket finns illustrerat i figurerna 3.3 och 3.4. Då kan man vara tillräckligt säker på att ha hittat början på ett transportströmpaket. Samtidigt ändrades Player-servletens buffertstorlek från 4096 tecken till 3948 (21*188) tecken i syfte att läsa hela TS-paket.



Figur 3.3: Funktionen `syncTS` säkerställer att transportströmmen är synkroniserad



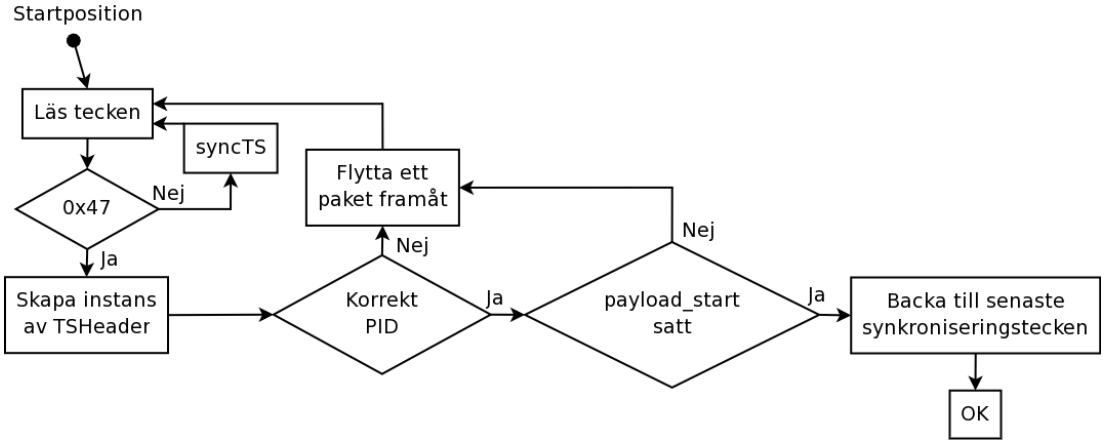
Figur 3.4: Flödesschema för `syncTS`

Detta implementerades som en funktion `static void syncTS (RandomAccessFile TS)`. Funktionen kommer att synkronisera den givna `RandomAccessFile TS` och avsluta. Även om detta inte kan förväntas innebära någon stor förbättring kommer funktionen att behövas vid implementationen av efterföljande förslag.

Resultat: Som förväntat gjorde detta ingen märkbar skillnad. Uppspelningen visar fortfarande lika mycket blockfel vid förflyttning. Detta är en följd av att mottagaren får fel datastrukturer för MPEG-strömmen, även om transportströmmen är korrekt. Fortfarande visar mediaspelaren i Kreatels mottagare tendens att krascha vid förflyttning.

3.1.3.2 Förslag 2 för störningsfri flyttning – paketerade elementära strömmens nivå

Förslag 2 bygger vidare på förslag 1, genom att man även här först måste synkronisera läsningen av strömmen på närmaste transportströmpaket.



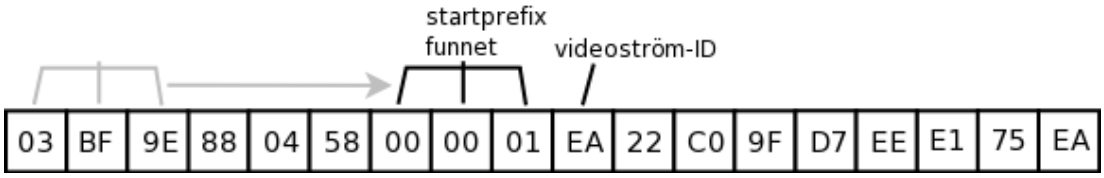
Figur 3.5: Flödesschema för `syncPayloadStart`

Funktionaliteten är implementerad i funktionen `static void syncPayloadStart (RandomAccessFile TS, int PID)`. Som figur 3.5 visar, kommer denna funktion efter att ha kallat på `syncTS`, att traversera transportström-filen paket för paket, instantierar `TSHeader`-objekt, tills ett paket med rätt PID och payload_start-flaggan satt hittas. Därefter avslutar `syncPayloadStart`, efter att ha backat `RandomAccessFile`-objektets läsposition tillbaka till synkroniseringstecket.

Resultat: Genom att se till att fortsätta på ett transportströmpaket innehållande början av ett PES-paket märktes en noterbar skillnad, vad gäller blockfelen. Dock har mottagaren, såväl som VLC, fortfarande en tendens att stanna upp, samt ibland krascha helt. Utifrån set-top-boxens loggmeddelanden, “PTS out of sync” kan antas att detta beror på den signifikanta förändringen i paketerade elementära strömmens tidsinformation, PTS och DTS, samt möjligen transportströmmens tidsinformation, PCR. Utöver detta meddelar mottagaren att transportströmmens kontinuitetsräknare är felaktig, vilket stämmer. Vad som inte är känt är vad som händer med de transportströmpaket som har felaktigt värde – används de oberoende eller ignoreras de?

3.1.3.3 Förslag 3 för störningsfri flyttning – elementära strömmens nivå

Förslag 3 bygger i sin tur vidare på förslag 2; fortfarande söker man upp ett paket med payload_start, men här undersöker man vidare paketets nyttolast. Funktionen `static boolean validStartPES (byte [] data)` returnerar true om teckenarrayen data innehåller videoström-, sekvens- och GOP-header. Dessa tre headers börjar alla med PES- och ES-startprefixet, 0x000001, följt av 0xE0-0xEF för videoström-header, 0xB3 för sekvensheader samt 0xB8 för GOP-header. Figur 3.6 visar hur funktionen söker genom teckenarrayen efter startprefix med efterföljande header-indikator.



Figur 3.6: `validStartPES` söker upp startprefixet och undersöker sekvens-header

Denna funktion kommer att köras från `syncPayloadStart` efter att ett transportströmpaket med payload_start-flaggan satt hittats, för att bestämma om det är lämpligt att starta från.

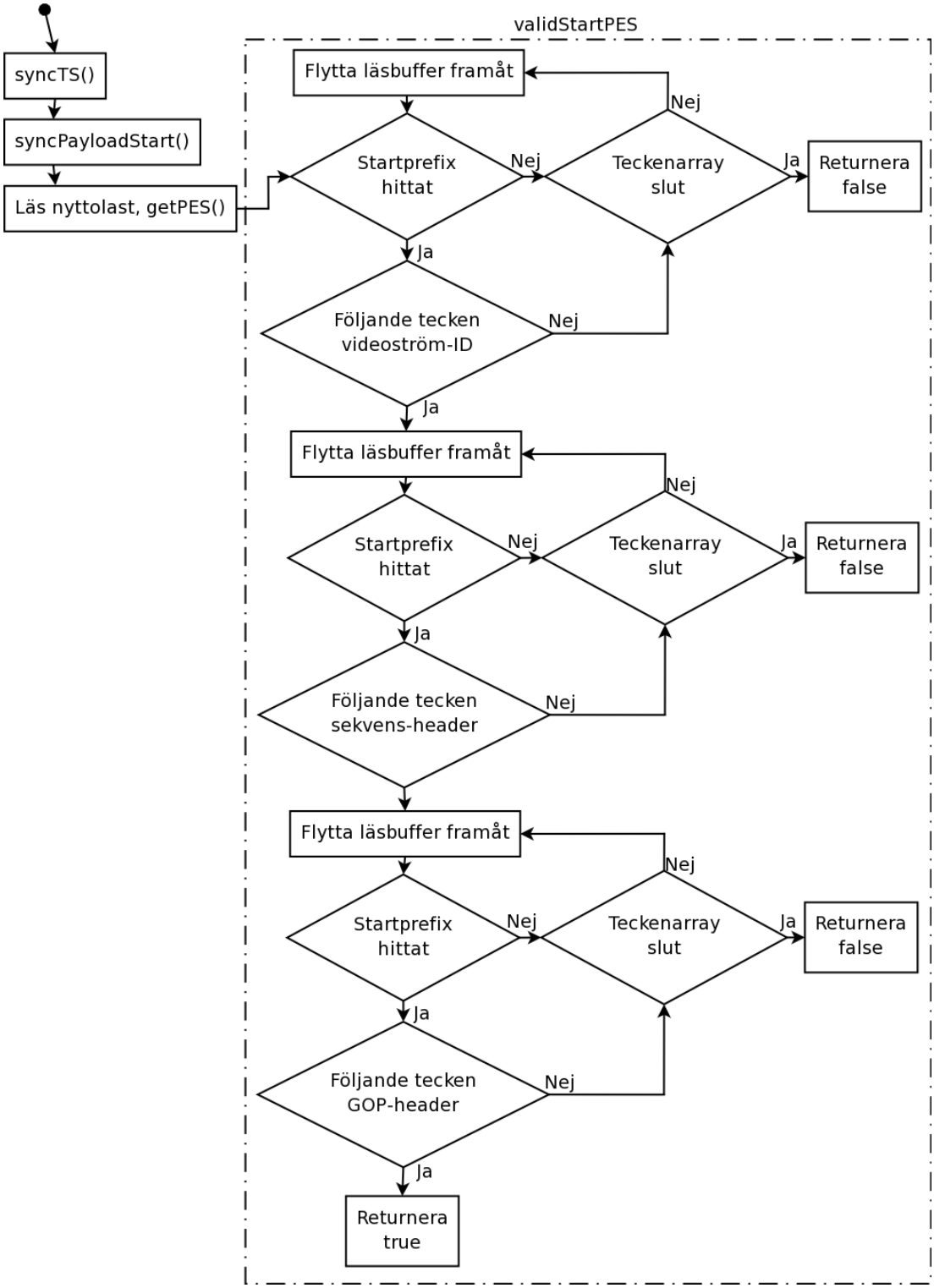
Funktionen `validStartPES` upprätthåller en form av tillståndsmaskin, illustrerad i figur 3.7, som kommer att inkrementeras för varje steg i identifieringen av ett korrekt paket. Om videoström-, sekvens- och GOP-header (i denna ordning) hittas kommer den booleska funktionen att returnera true, i övrigt false.

Resultat: Funktionen visar ingen förbättring i jämförelse med förslag 2, utan snarare en försämring. Genom att spara ner videoströmmen till en fil ser man dock att strukturerna är de förväntade, men även denna preparerade fil ger samma resultat. Problemen kan antas bero på att PCR-informationen i transportströmfilen samt PTS/DTS på PES-nivå är osammanhängande, vilket får både Kreatel-mottagaren och VLC att tappa synkroniseringen.

3.1.3.4 Föreslagen metod för tidsrefererad förflyttning

Player-servleten utökades med GET-parametrarna `skipms` och `seekms` och tillhörande metoder. Utöver detta sattes parametern `mseconds` till för att kunna överföra önskat antal millisekunder, ett värde som för `skipms` kan vara negativt.
Exempelvis en förfrågan

```
http://servername/player/Player?session=1&recording=37996
↪ &action=skipms&mseconds=-5000
```



Figur 3.7: Flödesschema över hur `validStartPES` bestämmer om ett TS-paket är lämpligt att starta från

kommer att förflytta den aktuella uppspelningssessionen bakåt med ca fem sekunder. När `skipms` sätts som värde på `action`-parametern kommer läsningen från transportströmfilen att stoppas. Därefter söker funktionen upp följande PCR-referens, skapar ett `PCR`-objekt och lagrar informationen. Utgående från `mseconds`-parameterns tecken, börjar funktionen flytta transportströmfilens position framåt eller bakåt, skapandes `PCR`-objekt, tills det beräknade slutliga PCR-värdet uppnåtts, enligt ekvation 3.2 och relativt till utgångspositionen. Slutligen körs `syncPayloadStart` för att fortsätta från ett giltigt transportströmpaket och läsningen startas igen. Figur 3.8 visar ett exempel i fallet att `skipms` skall söka framåt i transportströmfilen.

I fallet med `seekms` är `mseconds`-parametern relativ till början av filen. Funktionen börjar med att flytta transportströmfilens position till noll, och söker därefter upp första PCR-referensen. I övrigt analogt med `skipms` enligt figur 3.8. Nästa steg är att flytta framåt och skapa `PCR`-objekt tills önskat PCR-värde uppnåtts. Slutligen körs även här `syncPayloadStart` och läsningen tillåts fortsätta.

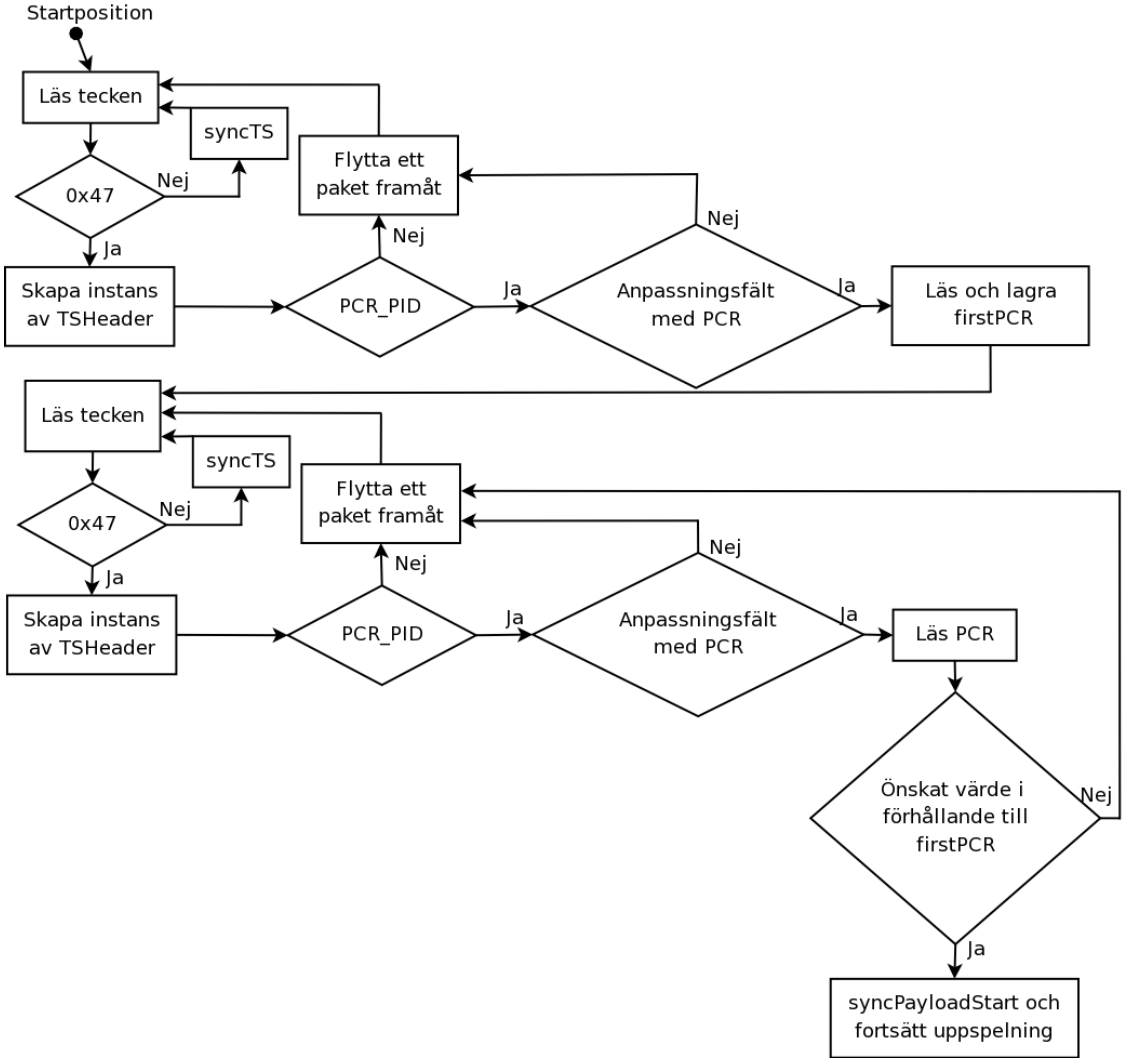
Resultat: För att testa funktionen ändrades klientmjukvaran till att kalla på Player-servleten med `skipms`-parametern och ett fast tidsvärde på 30 sekunder. Genom att spela upp en inspelning med synlig klocka kunde man bekräfta att funktionen för detta ändamål är tillräckligt noggrann; förflyttningen är ner till en sekund korrekt.

3.1.4 Ytterligare undersökningar av annan programvara

3.1.4.1 Anevia VOD-server

Hibox Systems har en VOD-serverlösning från Anevia[1] som möjliggör generering av multicast-transportström i realtid från lagrade mediafiler. Ett försök gjordes med att sätta igång en ström och ansluta till den med Kreatels set-top-box. Därefter ställdes serverns klocka om; vid omställning fem sekunder framåt stannar set-top-boxen upp i ca fem sekunder och fortsätter sedan uppspelningen. När klockan ställdes 45 sekunder bakåt stannar mottagaren också upp, och servern avslutar efter en stund med felmeddelandet “Unknown session”, vilket antas betyda att den inte längre har kontakt med mottagaren. I båda fall ger mottagaren samma felmeddelanden som med implementationsförslagen; “PTS out of sync” som en antydan om att den inte klarar av att synkronisera klockan till de nya tidsreferenserna.

[1] http://www.anevia.com



Figur 3.8: Flödesschema, exempel på hur `skipms`-funktionen flyttar uppspelningspositionen en bestämd tid framåt

3.1.4.2 VideoReDo

VideoReDo[2] är ett Windowsprogram specialiserat på redigering av MPEG. VideoReDo stöder klippning och ihopfogning av såväl transportströmmar som programströmmar.

Ett försök gjordes att foga ihop två urklippta bitar från exempelströmmarna på vardera 15 megabyte. Den resulterande transportströmmen fungerar felfritt att spela upp med såväl VLC som Kreatels IPTV-mottagare. Samma hopfogning gjordes på identiska GOP-positioner, manuellt, med en hexeditor. Denna fil hade dock samma problem som förslag 3 – VLC och IPTV-mottagaren stannar upp och ger diagnostikmeddelande om felaktig PTS.

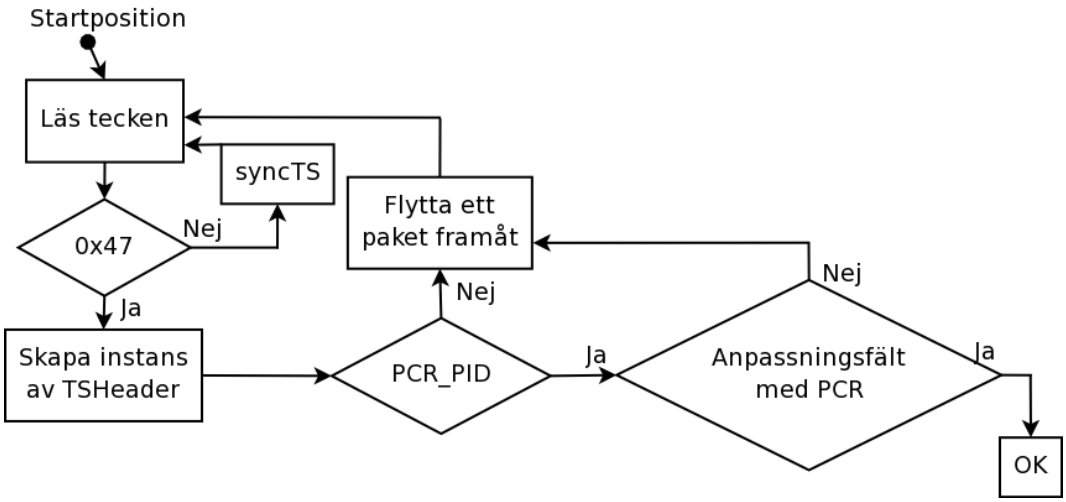
Vid undersökning av den resulterande transportströmfilen kunde följande konstateras; utöver att ha lagt PCR-information som anpassningsfält i videons PID (från att ursprungligen ha haft en fristående PCR-PID), har programmet generat ny PCR-information samt på PES-nivå, ny PTS/DTS-information. I detta fall har PCR-informationens upplösning också minskats till ca 90 ms från ursprungliga ca 38 ms. Den nya genererade PTS/DTS-informationen förekommer däremot signifikant oftare än i den ursprungliga strömmen. I övrigt har videoströmmens datastrukturer inte kodats om eller förändrats. Programmet använder sig inte heller av transportströmmens diskontinuitetsflagga. Korrekt tidsinformation kan därmed antas ytterst viktigt för att få en problemfri förflyttning.

3.1.5 Slutlig implementation på Hibox Systems server

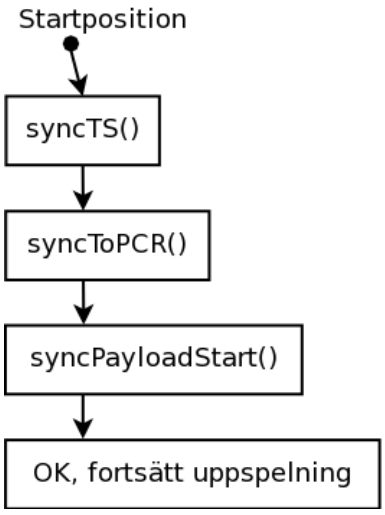
Vid testkörningen av den tidsrefererade sökningen observerades att MPEG-dekodern inte verkade stanna upp lika mycket som med implementationsförslagen som skulle lösa blockfels- och låsningstendenserna. Skillnaden i den tidsrefererade sökningen är att före `syncPayloadStart` körs kommer `RandomAccessFile`-objektets läsposition att stå på ett transportströmpaket som innehåller PCR-information. Således kommer uppspelningen att fortsätta från närmaste efterföljande transportströmpaket innehållande videoström-, sekvens- och GOP-header. För att närmare undersöka detta gjordes funktionen `static void syncToPCR (RandomAccessFile TS, int PCR_PID)`. Funktionen, som finns illustrerad med flödesschema i figur 3.9, utför i fallen `skip` och `seek` samma operationer som `skipms` och `seekms` för att söka upp ett paket innehållande PCR-information. Detta kan anses vara bästa möjliga övergång som kan åstadkommas utan att ingående förändra transportströmmens datastrukturer och tidsinformation. Hela förfarandet efter förflyttning i fallen `skip` och `seek` finns illustrerat i figur 3.10.

Resultatet kan förklaras med att MPEG-avkodarna i VLC och Kreatels set-top-box tolererar hopp i tidssinformationen bättre ju längre tid den har att anpassa sig före nästa

[2]http://www.videoredo.com



Figur 3.9: Flödesschema över funktionen `syncToPCR` som används vid icke-tidsrefererad förflyttning



Figur 3.10: Fullständig lösning för att hitta fortsättningsposition efter förflyttning

PCR-information. PTS- och DTS-fält återfinns dock omedelbart i videoström-headern och detta kan antas hjälpa till för att svänga in synkroniseringen.

3.2 Klientsida

Även om en stor del av systemet är serverbaserat kommer det att krävas förändringar i implementationen av den s.k. middleware-mjukvaran, dvs. den HTML-baserade portal som utgör IPTV-mottagarens användargränssnitt. Det primära är att lägga till funktioner för att kunna kalla på den tidsrefererade förflyttningen.

3.2.1 Kreatels IPTV-mottagare

För denna implementation används Kreatel/Motorolas IPTV-set-top-box, eftersom detta är leverantören som Hibox Systems huvudsakligen använder sig av. Kreatel är ursprungligen ett svenskt företag som idag ägs av Motorola. En artikel på Wikipedia [20] nämner Kreatels första IPTV-mottagare som lanserades år 2000, en lösning som senare såldes i betydande mängd till en italiensk bredbandsleverantör. I en artikel på The Register [61] berättas vidare om affären som ägde rum i början av 2006. Det genomgående budskapet i artikeln är Motorolas strategi för att komma in på en lukrativ IPTV-marknad genom att köpa upp en relativt välkänd specialist som Kreatel.

Gemensamt för Kreatel/Motorolas IPTV-set-top-boxar är att de alla är Linuxbaserade, har ett Ethernet-interface för anslutningen och minst 128 megabyte minne. Vissa modeller har även hårddisk och mottagare för DVB-signaler. Set-top-boxarna laddar sitt operativsystem över nätverket och för detta ändamål finns en speciell serverprogramvara.

Den s.k. middleware-programvaran som utgör största delen av IPTV-systemets användargränssnitt är uppbyggd av HTML och JavaScript. Detta portalsystem utvecklas av IPTV-leverantören, i detta fall Hibox Systems, och anpassas ofta enligt kundens önskemål med t.ex. logotyper. Portalsystemet skall finnas tillgängligt på en vanlig HTTP-server som t.ex. Apache HTTP Server eller Apache Tomcat, varifrån IPTV-mottagaren sedan laddar ner det.

För denna implementation har använts Motorola IP-STB 1510, figur 3.11, som enligt tillverkarens specifikationer [62] har en processor med 420 DMIPS beräkningsprestanda och 128 megabyte internminne. Enda ingång är ett RJ45-uttag för 10/100 Mbps Ethernet. Utgångar som finns är SCART, kompositvideo, s-video, två kanaler analog audio samt S/PDIF för digitalt 5.1-surroundljud. Anslutningarna visas i figur 3.12. Utöver detta har mottagaren en kortläsare för Conax-baserade kanalkort samt USB-gränssnitt för att t.ex. visa bilder från USB-minne.

Mottagaren, som är Linuxbaserad, saknar internt icke-flyktigt minne. Hela operativsystemet måste laddas över nätverket vid uppstart, och detta kräver att Kreatels serverprogramvara finns på en dator i nätverket. Serverprogramvaran använder UDP/IP multicast för att konstant sända ut de systempaket som finns definierade. Olika modeller av

Figur 3.11: Kreatels IPTV-mottagare tillsammans med fjärrkontroll.

Figur 3.12: Baksidan av Kreatels IPTV-mottagare.

mottagare har olika systempaket och därför kan flera definieras, tillsammans med bithastighet för utsändningen.

3.2.2 Mottagarens programmeringsmiljö

Kreatel-mottagaren använder HTML och JavaScript för att bygga upp sitt användargränssnitt. Utöver standard JavaScript finns funktioner för att kontrollera mediaspelarprogramvaran, dvs. starta och stoppa uppspelningen samt ändra hastighet vid uppspelning över RTSP-anslutningar. Detta är åtkomligt via ett MediaPlayer-objekt. Med hjälp av JavaScriptfunktioner kan man på klienten anknyta funktioner till olika knapptryckningar på fjärrkontrollen.

3.2.2.1 Teknik för kommunikation med servern

För att kunna kommunicera med servern utan att öppna en URL i den interna webläsaren används en teknik som har blivit känd som AJAX (Asynchronous JavaScript and XML). En artikel på Wikipedia [63] förklarar att AJAX möjliggör nedladdning av data i bakgrunden utan att ladda om en websida, vilket kan utnyttjas för att få bättre prestanda. En `XMLHttpRequest` är det centrala i AJAX, och används på klienten i ett skriptspråk som t.ex. JavaScript för att kommunicera med webservern. Detta skriptspråk kan sedan använda DOM (Document Object Model) tillsammans med HTML/XHTML och CSS (Cascading Style Sheets) för att visa informationen. Google anammade AJAX i ett tidigt skede och använder rikligt dessa tankesätt i sina tjänster Google Maps och Google Mail. Ett exempel på användning i mottagarens användargränssnitt:

```
function pauseServer() {
    var httpRequest = new XMLHttpRequest();
    httpRequest.onreadystatechange = function(){
        httpRequest = null;
    };
    httpRequest.open('GET',document.streams.sel.value+'&action=pause-resume',true);
    httpRequest.send(null);
    if (serverplaying == 0) serverplaying = 1;
    else serverplaying = 0;
}
```

Funktionen `pauseServer()` kommer att kallas på när man trycker på Play-knappen och en uppspelning pågår. `XMLHttpRequest`-objektet kallar sedan på servern med GET-parametern `action` som har värdet `pause-resume`. I detta fall kommer `document.streams.sel.value` från en intern meny och innehåller resten av serverns URL.

3.2.3 Implementation

En exempelportal har utvecklats med några strömmar som kan spelas upp, för att visa hur funktionerna kan användas. Denna har baserats på Kreatels exempel samt Hibox Systems middleware. Eftersom MediaPlayer-objektet inte kan kommunicera med denna specifika servlet på samma sätt som med en RTSP-server, måste funktioner sättas till runt om.

Det centrala är att läsa in tiden för förflyttningen och detta knyts till händelser från fjärrkontrollen. När användaren trycker på knappen för att spola framåt respektive bakåt, kommer en variabel att inkrementeras respektive dekrementeras med 10000 millisekunder. Samtidigt startas en timer på en sekund som räknar ner och slutligen gör förfrågan till servern, om inga flera knapptryckningar tagits emot. Om man däremot trycker flera gånger nollställs timern och börjar från utgångsvärdet igen.

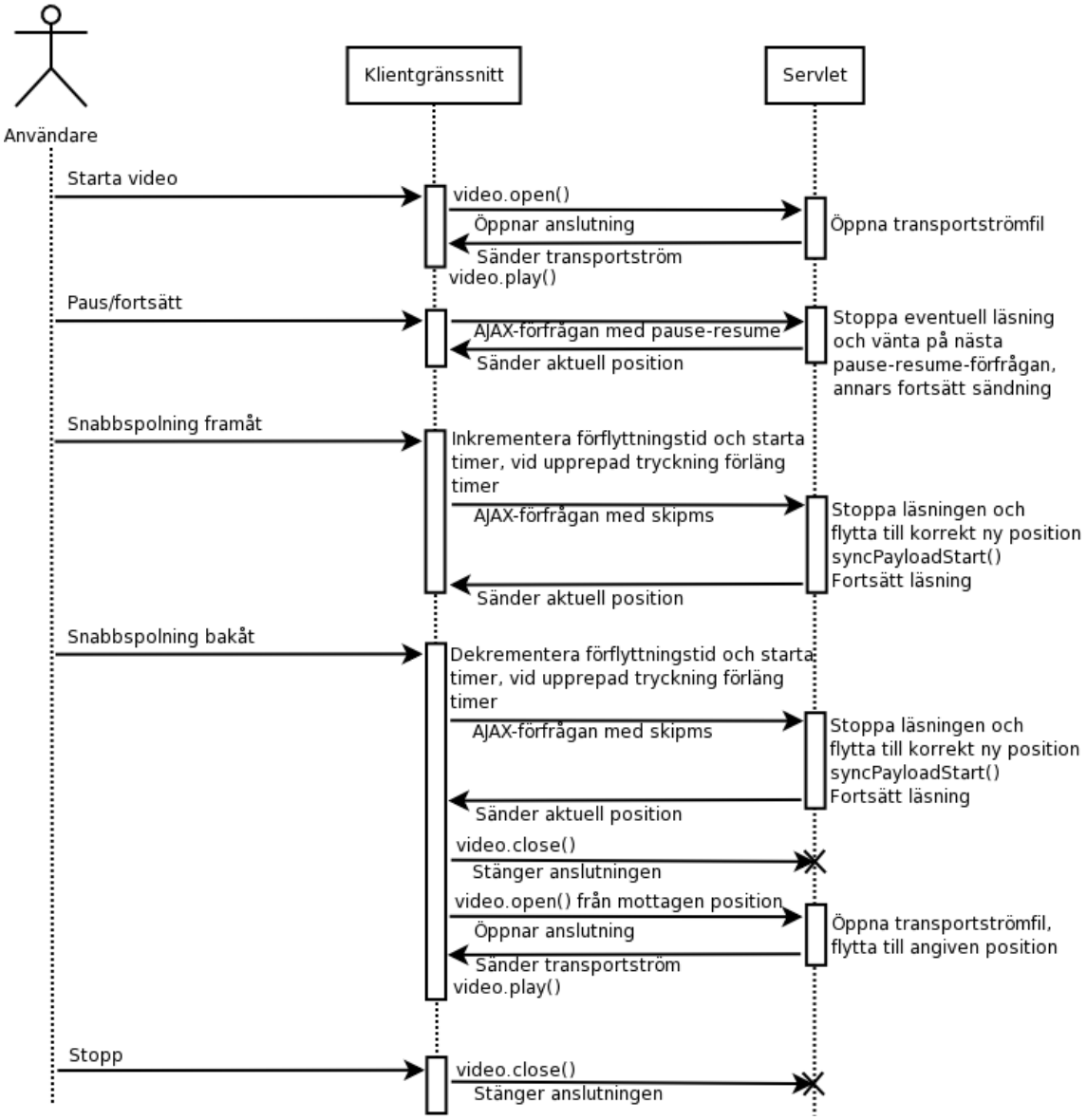
I fallet med hopp framåt bör det vara tillräckligt att göra en AJAX-förfrågan till servleten med `skipms`-parameter. Däremot vid bakåthopp (vilket är förbjudet enligt ISO/IEC 13818-1 [44]) måste ytterligare åtgärder vidtas eftersom mottagaren annars har tendens att krascha. Servletens normala funktion är att sända ut positionen i transportströmfilen som svar på varje förfrågan, vilket i detta fall innebär början av ett transportpaket med ny GOP. Genom att starta om uppspelningen och utnyttja denna position fås felfri förflyttning bakåt. Funktionen är följande:

```
function rewPressed() {
   try {
      var status = video.getState(); //mediaspelarobjektets tillstånd

      if (status == video.STATE_PLAYING){
         if (serverplaying == 1) { //servern sänder data
            httpRequest = new XMLHttpRequest();
            httpRequest.onreadystatechange = function() { //körs när respons mottagits
               if (httpRequest.readyState == 4) {
                  if (httpRequest.status == 200) {
                     var i = 0;
                     i = parseInt(httpRequest.responseText); //läser aktuell position
                     video.close(); //avslutar uppspelningen
                     sleep(1000); //väntar så servleten hinner ta bort strukturer
                     video.open(document.streams.sel.value+'&position='+i,);
                     video.play(1000,-1,-1); //öppnar ny anslutning och spelar video
                  }
               }
               httpRequest = null; //frigör httpRequest-objektet
            };
            currentjumps-=10000; //tio sekunder bakåt per knapptryckning

            if ( clearJumpsThread ) { //avbryter befintlig nedräkning
```

```
                clearTimeout( clearJumpsThread );
                clearJumpsThread = null;
            }
            //för visning i användargränssnittet:
            document.getElementById(”jump”).firstChild.data = currentjumps;
            //startar timer som kallar på resetCurrentJumps() efter en sekund:
            clearJumpsThread = setTimeout( ”resetCurrentJumps()”, 1000);
         }
      }
   }
   catch (e) { }
}

function resetCurrentJumps() {
   //skapar förfrågan med httpRequest-objektet
   httpRequest.open('GET', document.streams.sel.value+'&action=skipms&mseconds='+
   ↪ currentjumps, true);
   httpRequest.send(null); //och sänder den
   currentjumps = 0;
   document.getElementById(”jump”).firstChild.data = 0;
}
```

`XMLHttpRequest`-objektet `httpRequest` har gjorts som en global variabel som används av alla metoder. Funktionen `rewPressed()` kommer att kallas på när fjärrkontrollknappen för bakåtspolning trycks ned. Funktionen tilldelar en metod som körs vid `httpRequest.onreadystatechange`, dvs. när servern sänt respons, och därefter startas timern som efter en sekund kallar på `resetCurrentJumps()` som i sin tur gör förfrågan till servern. Slutligen när servleten har gjort sina uppgifter och sänder respons stoppar klienten uppspelningen och öppnar en ny anslutning från den mottagna positionen, med `position`-parametern.

Figur 3.13 visar ett sekvensdiagram för alla kommandon från användarens fjärrkontroll.



Figur 3.13: Sekvensdiagram för kommandon från användaren genom mottagarens fjärr-kontroll

Kapitel 4

Diskussion, resultat

Det huvudsakliga målet med mitt diplomarbete har varit att undersöka och om möjligt lösa problemet med blockfel och låsning vid förflyttning (realtidssammanfogning) i MPEG-transportströmmar. Den sekundära uppgiften var att kunna förflytta positionen ett önskat antal millisekunder istället för bara antal tecken. Dessa kravspecifikationer visade sig senare vara ganska orelaterade eftersom de befinner sig på olika nivåer i transportströmmen.

Efter teoretiska efterforskningar började jag se att problemet med förflyttning i transportströmmen skulle vara tekniskt lösbart, även om det kräver ingående analys av MPEG-formatets datastrukturer. Det avancerade MPEG-formatet är uppdelat i flera skikt och därför började jag systematiskt från de högre lagren och gick neråt mot lägre nivåer. Genom att utveckla ett antal verktyg för att läsa in dessa strukturer, kunde en användbar algoritm utformas. Metoderna är inte beräknings- eller minnesmässigt krävande, däremot kan diskaktiviteten utgöra en belastning om många klienter samtidigt utnyttjar förflyttningsfunktionen.

Metoden för att flytta position önskad tid framåt eller bakåt fungerar mycket bra – förflyttningen är ner till en sekund korrekt. I synnerhet här kan diskaktiviteten visa sig vara ett problem, och en möjlig förbättring på serversidan kunde därför vara att estimera hur långt man inledningsvis skall förflytta läspositionen. Det blir ineffektivt att läsa genom stora delar av filen vid långa hopp. Vid hopp bakåt är det nödvändigt att stoppa mottagarens mediaspelare (understöds inte av MPEG), men även vid förflyttning framåt kan det vara en bra idé, eftersom set-top-boxens buffertminne i båda fallen annars orsakar en fördröjning på ca två sekunder.

Ett problem om man inte stoppar mottagarens mediaspelare är även att mottagaren har tendens att stanna upp ungefär en sekund efter förflyttning. Detta fenomen uppstår på grund av diskontinuiteten i tidsinformationen, vilket leder till att mottagarens MPEG-avkodare väljer att visa stillbild tills den synkroniserat efter den nya tidsinformationen. Det går inte på enkelt sätt att åstadkomma en förflyttning utan diskontinuitet. Däremot om man från klientsidan stoppar mediaspelaren och startar den igen från den nya po-

sitionen åstadkoms en acceptabel start. Givetvis är det helt möjligt att göra en sömlös hopfogning i realtid, men det kräver att man implementerar en lösning som genererar ny tidsinformation för materialet.

Även om den störningsfria realtidsförflyttningen skulle visa sig vara mera komplicerad än jag från början kunde tro är det alltså möjligt att bygga sig runt dessa svårigheter genom att anpassa klienten. Man kan slutligen säga att jag har uppnått målen med diplomarbetet och åstadkommit en realistisk lösning som går att använda på ett produktionssystem.

Bilaga A

Summary in English

A.1 Introduction

Perhaps the biggest change in television technology is taking place at the moment with the transition to digital systems. All over the world, legacy analog transmissions are being switched off and replaced by digital television such as DVB and ATSC. One reason for doing this is the crowded radio frequency spectrum — with digital compression it is possible to convey several channels in the bandwidth of one analog channel. Another reason is the potential for higher quality reception and HDTV (High Definition Television).

A.1.1 The concept of multimedia over computer networks

The idea for transmitting media over computer networks and the Internet took hold as quickly as enthusiasts started interconnecting their computers. In the mid-nineties, the concept of Internet radio emerged (one pioneer mentioned on Wikipedia [9] was Carl Malamud in 1993, utilizing the MBONE multicast network), and soon video was added to form Internet television. The first actual Internet television station mentioned by Wikipedia [10] was World News Now from ABC in 1994. A number of services that can be considered technological and legal "grey zones" have turned up, including YouTube[1] and TVU networks[2], where sometimes copyrighted material can be illegally viewed online. One new concept is Joost[3], formerly known as The Venice Project. An article on BusinessWeek by Steve Rosenbush [12] explains this latest idea by Niklas Zennström and Janus Friis, the creators of the file sharing system Kazaa and the Internet telephony system Skype. They intend to have a close cooperation with the channels that are to be transmitted, and the system is based on peer-to-peer technology.

[1] http://www.youtube.com
[2] http://tvunetworks.com
[3] http://www.joost.com

A.1.2 The IPTV concept

One technology that has emerged as a result of fast and affordable home data networks ranging from DSL to optical fiber, is IPTV or Internet Protocol Television. Earlier, cable network operators were in a monopoly position area-wise as they owned the cabling. Now, with IPTV over Internet connections, new possibilities for competition are opening up since IPTV can function independently of the Internet provider.

A report published by Broadband Services Forum called “IPTV Explained” [13] describes the nature of an IPTV system. The well-known IP protocol is used, commonly together with a transport protocol such as UDP for the data transmission. By using an IP network, there will already be a return channel available, which is necessary for many interactive services.

Since a standard IP infrastructure is available, the possibilities for additional services besides television and VOD (Video On Demand) are wide. Broadband Services Forum shows some examples in the report “The IPTV Bundle” [14] — music subscriptions, video conferences, home video surveillance and online dating are lucrative concepts already deployed on the Internet.

Even though IPTV shares some characteristics with streaming video over the Internet, it is important to differentiate IPTV from Internet TV. Broadband Services Forum sums up the differences and similarities in the report “IPTV Versus Internet TV” [15]. An IPTV network is often limited to a smaller local network — for example a hotel — instead of having the transmissions on the Internet. This helps to ensure the robustness and quality of service since the network is privately managed and limited. Another point is that IPTV should feel like television — flawless, responsive and everything controlled with the remote control.

A.1.3 IPTV systems today

The server side plays a major role in an IPTV system. The server side tasks described by Broadband Services Forum in “IPTV Explained” [13] includes receiving digital television transmissions (a DVB transport stream), separating the programs and feeding them into the network as a stream of raw UDP packets or RTP over UDP. Generally, IP multicast is used for real-time content. The advantages of this according to the report are that a wide variety of video material can be distributed to a large number of clients, while the utilized bandwidth is kept at an acceptable level.

Video on demand is a feature usually provided by IPTV operators. A VOD server is often bought as a commercially made solution, but still requiring the operator to implement functions on the server side for access control and billing.

Most of what is visible to the user from the client side is downloaded from the ser-

ver, generally in XML form with background and pictures. This user environment may contain functions such as EPG (Electronic Program Guide), weather forecasts, news and interactive games. Finally, a common function is PVR (Personal Video Recorder) which enables the user to record content to view at a later point — similar to a VCR, but the recordings are stored on the operators server and no extra appliance is needed. This server software is generally refered to as middleware and is described by Broadband Services Forum [13] as a portal that links all the components together. Due to the lack of standards for IPTV receivers, developing the middleware in-house is usually the best solution.

The client system, a specialized set-top box for IPTV, does not generally have an RF tuner, only an Ethernet interface. It is an embedded system with a general processor which is powerful enought for its tasks, a relatively small amount of RAM and sometimes a non-volatile storage for storing the operating system when being turned off. On startup, the receiver will load its operating system over the network from the server, but will be able to start almost instantly if it is stored locally. The operating system is remote-updateable from the server. The receiver has a remote control and sometimes an optional wireless keyboard. A simple web browser is also often included.

There are a handful of dominating manufacturers of IPTV set-top boxes, inluding Motorola[4], Kreatel[5] — originally a Swedish IPTV pioneer, but according to a Wikipedia entry [20], nowadays owned by Motorola. Tilgin[6] — another Swedish company, Norwegian Tandberg[7], the Brittish Amino[8] and the French Thomson[9].

A.1.4 Hibox Systems' layout and the presented problem

Hibox Systems, founded in 2005, is a company in Turku specializing in IPTV solutions both for hotels and housing networks. The server side system is developed by Hibox Systems themselves and is based on Java servlets, runs on Apache Tomcat[10] and uses a MySQL[11] database. This enables the portal user interface to be very dynamic. Most administrative functions can be done in a web interface, such as defining which receivers have the right to access certain functionalities. A ready-made VOD server (hardware aswell as software) from BitBand[12] is used.

IPTV terminal chosen is IP-STB series from Kreatel. The MIPS based terminals run

[4] http://www.motorola.com
[5] http://www.kreatel.com
[6] http://www.tilgin.com
[7] http://www.tandberg.net
[8] http://www.aminocom.com
[9] http://www.thomson.net
[10] http://tomcat.apache.org
[11] http://www.mysql.com
[12] http://www.bitband.com

a Linux[13] based operating system with an X11[14] implementation for the graphical functions, and the Mozilla[15] web browser.

Hibox Systems has implemented a PVR function in their middleware, to allow users to record a program simply by selecting it in the EPG and adding it to a recording list. The stored content can be viewed later anytime the user wants, making this function a replacement for a VCR. This recording function works well, as does the playback, which uses HTTP to transfer the video. The problem consists of moving forward or backward in the video stream on user command. A simple test implementation has been made that moves the read position a number of bytes, not taking into account the data structures of the media. The result is that the receiver gets confused due to unexpected data structures, leading to corrupted video. This is due to the MPEG format with I, B and P frames forming groups of pictures (GOP). In order to get a smooth transition, it will be necessary find the right position to continue playback from. It may be necessary to index the material in advance in order to quickly find the right positions. The goal of this thesis project is thereby to investigate if it is possible to improve the fast forward/backward, and possibly implement a functionality for the purpose.

Besides for the smooth transitioning functionality, Hibox Systems wishes a function to move the playback position a number of seconds instead of bytes. Since MPEG does not have a constant bit rate, it will be necessary to read time stamps from the stream and calculate where to continue playback.

A.2 The MPEG standard and DVB

MPEG (Moving Picture Experts Group) is more than just a collection of video formats. It is a group of experts and industry key persons that designs standards for audio and video compression. According the the official MPEG homepage [22], the group was founded in 1988 and originally consisted of 25 members.

A.2.1 About the standards

MPEG has produced a number of standards. The most relevant mentioned on the homepage [22] are the ones definining video and audio: MPEG-1, MPEG-2 and MPEG-4, from 1989, 1991 and 1995 respectively. John Watkinson compares these in “The MPEG Handbook” [23]; MPEG-1, originally intended for the VideoCD format, was designed to have a bit rate of 1.5 MBit/s. This was in order to fit both audio and video on a CD and achive the same playtime as an uncompressed audio CD. MPEG-2 extends the simple but rather

[13] http://www.linux.org
[14] http://www.x.org
[15] http://www.mozilla.org

limited MPEG-1 with many new functions. Watkinson [23] points out the backward compatibility — an MPEG-2 decoder can handle all MPEG-1 material. One function added is interlacing — only every second line is drawn per frame, similarly to analog television. MPEG-2 consists of four standard resolutions, “levels”, and six “profiles” for different purposes — for example bandwidth, picture quality and backwards compatibility. Finally, MPEG-4 is the latest incarnation, existing in version 1 and 2. Watkinson [23] describes the new functions for modelling movements in much more dynamic ways than MPEG-1 and MPEG-2. Mobile objects can, besides moving and rotating two-dimensionally, also stretch and turn three-dimensionally. Furthermore, MPEG-4 has functions for face and figure animation, enabling video conferences over very low bandwidth. As with MPEG-2, MPEG-4 has a number of profiles for different purposes.

Watkinson [25] points out one fundamental principle of MPEG — nothing in the standard specifies how an encoder should be implemented. The part which is thoroughly specified, is the decoder. The decoder implementation is comparatively trivial, since it only needs to interpret a well-known bitstream, while the encoder is allowed to make complex decisions about the compression.

A.2.2 MPEG video compression

The main idea with MPEG is to eliminate redundant information in the material. There may be redundancy per frame — larger surfaces with the same color — aswell as redundancy between frames — many details don’t change between two frames. By removing unneccessary information it is possible to spare bandwidth and storage space.

Watkinson [27] describes the method for compression within a frame. The picture is divided into a number of 8x8 pixel blocks, which are processed by the discrete cosine transform (DCT). DCT transforms the material into the frequency domain, making it trivial to select the most significant frequencies and discard the rest. Quantizing the coefficients to discrete values is what does the actual compression. Four DCT blocks make one 16x16 pixel macroblock, macroblocks that finally build up the entire picture — for example, a resolution of 720x576 consists of 45x36 macroblocks — and this is what is referred to as an I frame, or intra frame.

MPEG will generally achieve good compression within frames, but nearby frames are often very similar — this is known as temporal redundancy — and MPEG utilizes this to improve compression by only conveying differences between frames. Watkinson [30] shows the methods used; besides sending the subtracted difference, MPEG applies a method with motion vectors to describe movement in the content. Motion vectors together with differential data is what makes a predicted frame, or P frame. Finally, it is possible

to use data from a future frame aswell as from earlier — bidirectionally predicted frames, or B frames.

The encoder generates a bitstream of I, P and B frames i a predefined order, called a Group Of Pictures. Watkinson [30] gives a sequence IBBPBBPBBPBB as a common structure, and a GOP will always begin with an I frame.

A.2.3 Three layers of streams

The audio and video bitstreams produced by the encoder have to be packetized properly before transfer. Watkinson [35] describes these in the chapter “Program and transport streams”. The encoder will output an elementary stream (ES), containing the groups of pictures mentioned earlier. The next level is the packetized elementary stream (PES), which splits the stream into manageable blocks together with timing information. On top of this, the transport stream (TS), which enables simultaneous transfer of several programs built up of video, audio and data.

The elementary video stream is specified by ISO/IEC in the 13818-2 [36] standard is raw output from the encoder only split up by start code and prefixes indicating start of pictures, slices and GOPs.

To simplify handling, the elementary stream is split up in packets forming a packetized elementary stream. Even if the size is quite liberal, Röjne [37] states that a PES packet usually contains one picture or 24 ms of audio. ISO/IEC 13818-1 [38] defines that a PES packet starts with the same prefix, 0x000001, as an ES structure, but the PES packet identifies for example video or audio streams. Timing information is partially handled on the PES level. Watkinson [35] describes the PTS (Presentation Time Stamp) and DTS (Decode Time Stamp) fields, which are used to synchronize display of video frames and audio. Both timestamps are 33 bits, sampled from the encoder clock which is incremented at 90 kHz.

The transport stream (TS) is the final packetizing and multiplexing. A transport stream can contain many packetized elementary streams. For example, at the moment the digital television transmissions in Finland consist of only three transport streams (Digita [39]). A transport stream packet is always 188 bytes, and each individual video, audio or data stream is identified by its 13 bit Packet Identification Code (PID) number. ISO 13818-1 [40] defines the transport stream packet format; it will always start with the synchronization byte 0x47, and can contain an adaptation field for additional information.

The system for transferring program information also resides at the transport stream level. This is called Program Specific Information or PSI. Watkinson [35] shows how it works; the Program Association Table (PAT) is sent out as PID zero and contains a list of all available programs (consisting of a video stream and one or more audio streams)

together with a PID for their Program Map Table (PMT). PMT in turn lists the PIDs of the video and audio streams belonging to the program.

The transport stream has a system for time synchronization supporting the PTS/DTS of the PES. Watkinson [35] points out the important concepts of PCR (Program Clock Reference). The PCR information will be contained in the transport stream packet adaptation field, and is used by the decoder to re-create the encoder clock at 27 MHz. This clock is necessary for maintaining synchronization of PES streams.

A.2.4 MPEG splicing

The MPEG standard, ISO 13818-1 [44], discusses this highly relevant subject in the chapter “Annex K: Splicing Transport Streams”. Two approaches are mentioned: seamless and non-seamless splicing. The difference lies in how the timing information is changed after the splice; seamless splicing will be completely transparent with regard to time stamps. Non-seamless splicing allows a significant jump in timing information leading to a discontinuity. Neither case allows a backwards jump in time stamps.

In this case with real-time splicing, non-seamless is the only realistic. According to ISO 13818-1 [44], the packet following a splice must contain PTS/DTS information. What can cause problems is how the decoder treats discontinuities — the standard suggests showing a few freezed frames.

A.3 The HTTP protocol

HTTP — HyperText Transfer Protocol — may be the most commonly used data communication protocol on the Internet today. Chris Shiflett tells the history behind HTTP in “HTTP Developer’s Handbook” [45]; the original draft, “Information Management: A Proposal”, was presented by Tim Berners-Lee in march 1989. Tim Berners-Lee, who can be considered one of the most influential persons in making the World Wide Web and the Internet into what it is today, presented the proposal to CERN (the European organization for nuclear research). The first implementation, HTTP 0.9, was only intended to transfer and present text, but the protocol has evolved from that. The latest version is HTTP 1.1 from 1999 and that is almost exclusively used today.

In the OSI model described by Shiflett [46], HTTP is on top, in the application level. Together with HTTP, only TCP (Transmission Control Protocol) is used at the transport level and IP (Internet Protocol) at the network level. TCP will ensure correct delivery of packets, and DNS (Domain Name System) is used together with IP to resolve IP addresses.

A.3.1 Requests and responses

Like many other high-level protocols, HTTP is entirely text based and legible. Even though text based protocols consume larger bandwidth than binary protocols, it is compensated by simplicity in implementation and troubleshooting. HTTP uses an address format called URI or URL (Uniform Resource Indentifier or Uniform Resource Locator) to identify a resource. According to Shifletts [46] description, a simplified example can be given:

```
http://ourserver.fi/subdir/index2.php?session=5
```

where `http` indicates the HTTP protocol, `ourserver.fi` is a domain name translated to the servers IP address by DNS, `subdir` is a subdirectory on the server. A filename is indicated as `index2.php` and a variable `session` with value 5 is transferred to the server. Usually, this is used by the file being fetched, for example to influence the content of dynamic web pages.

There are eight requests and almost 50 response codes, described in detail by Shiflett in the chapter "HTTP Requests" [47] in HTTP Developer's Handbook. In HTTP 1.1 all requests consist of a request line, a Host line and possibly one or more request specific headers, all terminated by CRLF (Carriage Return + Line Feed). Requests supported by HTTP 1.1 are as follows:

`GET`, `POST`, `PUT`, `DELETE`, `HEAD`, `TRACE`, `OPTIONS` and `CONNECT`

A response will be sent back from the server with an HTTP response code. This is similar to the request and Shiflett [48] describes them in detail in the chapter "HTTP Responses". There are five types of status codes as follows:

- Informative (100-199)
- Success (200-299)
- Redirection (300-399)
- Client error (400-499)
- Server error (500-599)

A.3.2 A stateless protocol

An important property which Shiflett [50] points out is the stateless nature of the HTTP protocol. This means that the protocol by no means attempts to keep track of earlier requests made by a client. All an HTTP server per definition does is to process individual

requests. A need has arisen to implement session management in HTTP with different methods. Cookies is one of them, a domain specific string stored by the client that can be used for identification and storing individual settings. GET and POST requests are another popular method. By using a script language such as PHP, a session specific string can be generated to keep track of sessions. This can be sent as a variable in the URL, as follows: (based on example by Shiflett [50]):

```
GET /index2.php?session=5 HTTP/1.1
Host: ourserver.fi
```

In this case, `index.php` may have contained a function that tracks current sessions, and assigned this client a session variable of 5.

A.3.3 HTTP and streaming media, Hibox Systems' solution

Streaming media over HTTP works just as with all other content — TCP guarantees in-order delivery. This has the potential to turn into a disadvantage on an unreliable channel. TCP will send the lost information again, while the receiver blocks until it has been received, possibly resulting in stopped playback. This is usually avoided by having large enough buffers in the receiver. For streaming media, UDP is otherwise usually used, which does not have error checking. A reason for this is that a lost packet will not cause much problems decoding media data.

Another issue in streaming media over HTTP may be the slow start algorithm of TCP. This is used to determine the network capacity and avoid congestion, by gradually enlarging packet size. With many ongoing connections, this can be a problem since the other TCP connections have to adapt their packet size and possibly re-send lost data.

Hibox Systems has implemented the PVR playback functionality over HTTP. To support this, the middleware player functionality is a Java servlet that gives full control over how the requests are handled and what is sent to the client. State management is handled with a `session` number assigned by the middleware. The client sends this as a GET variable in the requests. Another GET variable is the `recording`, which is associated with the appropriate file by the servlet. One example could be:

```
http://ourserver.fi/player/Player?session=2&recording=1357
```

which will make the server deliver the recording with ID 1357, and remember that the client with session 2 is playing it.

By adding a variable `action`, the playback is controlled. Valid values for this variable are `skip`, `seek` and `pause-resume`. `skip` requires the integer variable `bytes`, which may be positive or negative for moving backwards or forward in the stream. In the same way,

seek requires the position variable which moves the file pointer to an absolute position. Example of usage:

```
http://ourserver.fi/player/Player?session=2&recording=1357
↪ &action=skip&bytes=-20000
```

moves the playback 20000 bytes backwards. At this point, it will be necessary to find a proper position in the transport stream file to continue reading, since the current system continues directly. This unexpected change in data structures causes problems for the receiver, and results in block artefacts.

A.4 Implementation, server side

A.4.1 Initial observations

A number of tools were developed in order to analyze the transport stream samples. Java was a natural choice for all the development since Hibox Systems' server software is Java based. Common for all the tools is the use of a java.io.RandomAccessFile object for accessing the files. This allows binary reading/writing and random access. The very first tool, readts searches the file for the transport stream synchronization character (0x47) and instantiates an object of class TSheader. This object displays information about the packet, aswell as position, which is useful for finding the packet in a hex editor. A hex editor has proven very useful in understanding the MPEG structure.

In order to better understand the packetized elementary stream, the dumpvideoPES tool was developed. This writes the payload from the video PID, or any specified PID, to a file. The file is playable for example with the VLC player.

It would prove necessary to parse PSI (Program Specific Information), the system for describing the transport stream contents. Since the PVR transport streams contain only one program, only the PID of its video stream is needed. The parsing functionality is implemented in the PAT and PMT classes, which both take a RandomAccessFile object as constructor argument. The read position in this object has to stand at the correct position, in the beginning of the payload. The function static PMT findVideoPMT (RandomAccessFile TS) will return a PMT object for the first program. This PMT object contains two integer arrays with PID and their stream type. According to ISO/IEC 13818-1 [42], the video stream will have type two or three.

The PCR system was studied with the showPCR tool. It begins by calling the findVideoPMT function and reads the PCR PID from the received object. After that, it starts going through the transport stream file, packet by packet, and creating objects of class PCR if the packet contains an adaptation field. If the PCR flag in the adaptation field is set, the

object will parse PCR. The `showPCR` tool will then display information including current PCR time stamp and difference from previous in ticks and milliseconds in this form:

```
PCR info found at pos d7e8
Parsing adaptation field size 183
Discontinuity: 0
Random access: 0
ES Prio: 0
PCR: 1
OPCR: 0
Splicing point: 0
Private data: 0
Adaptation field extension: 0
PCR base field: 5213676333
PCR extension field: 108
Full PCR: 1564102900008

Difference in PCR: 1017144 ticks
In mseconds: 37
```

The `analyzePES` tool was made in order to examine the stream at PES level. It will search for the start code prefix and instantiate an object from a class corresponding to the start code. Implemented classes are `streamheader`, `sequenceheader`, `GOP` and `pictureheader`. Each of these objects will parse the data structures and display relevant information in this format:

```
Prefix found at pos: de1c: ea, identified as video stream header
PES packet length: 0
Scrambling: 0
Priority: 1
Alignment: 1
Copyright:1
Original/copy: 1
PTS/DTS: 3
ESCR flag: 0
ES rate: 0
DSM trick mode: 0
Additional copy info: 0
PES_CRC: 0
PES extension: 0
PES header length: 10
PTS: 5213798416
DTS: 5213787616

Prefix found at pos: de2f: b3, identified as sequence header
Horizontal size value: 720
Vertical size value: 576
Aspect ratio information: 2
Frame rate: 25
Bit rate: 25003
VBV Buffer size: 568

Prefix found at pos: de7b: b5, identified as extension header
Prefix found at pos: de85: b2
Prefix found at pos: de8f: b8, identified as GOP header
```

```
GOP with time 7:17:36
Frame 2, closed GOP: 0, broken: 0, next_start: 0

Prefix found at pos: de97: 0, identified as picture header
Temporal reference: 2
Frame type: I
VBV Delay: 65535
```

The observations made were as follows; at an early stage it became clear that the DVB transmissions never contain “splicing points” as a flag in the adaptation field. Something that instead would turn out to be relevant was the indicator payload_start. This indicates the start of payload, and the start codes for sequence and GOP only exist in transport stream packets with the payload_start flag. All packets with payload_start do not contain a new GOP though. A GOP start code is preceded by the elementary stream sequence start code, which is preceded by the video stream ID. By extracting the PES beginning from from a packet with GOP, a start without block artefacts is achieved.

Transport packet with the scrambling indicator flag set are lacking these properties. These Conax encrypted streams never have the payload_start flag set, and the payload never contains the start code prefix. It will not be possible to find a suitable start position without decoding them.

The resolution of the PCR information is between 25 and 40 milliseconds depending on stream, although constant per stream with a few milliseconds variations. At the PES level, PTS and DTS occur variably depending on stream, although before a GOP there will always be both PTS and DTS, which according to ISO/IEC 13818-1 [44] is a requirement for non-seamless splicing.

A.4.2 Possible methods for interference-free moving of playback position

Based on the previous observations, these possible solutions could be presented:

A.4.2.1 Method 1: transport stream level

The simplest solution would be to synchronize the transport stream to a complete transport stream packet near the new position. The last packet before the jump should also be sent in its entirety, in order to avoid confusing the receiver with erroneus and unexpected data. This can not be expected to significantly lower the level of block artefacts.

A.4.2.2 Method 2: packetized elementary stream level

Not much more complicated than method 1; this means searching for a transport stream packet with the payload_start flag set. This will lower the amount of block artefacts, but

not remove them completely since not all these packets contain the combination of stream-sequence-GOP headers.

A.4.2.3 Method 3: elementary stream level

This solution builds further on method 2, but this involves searching through the packet to see if it contains the correct combination of headers. By starting from a transport stream packet containing stream-sequence-GOP headers, the playback should be flawless.

One issue may be how the receiver interprets inconsistensies in the PCR, PTS and DTS. The worst case would be that it ignores packets with unexpected time stamps or shows freezed frames as ISO/IEC 13818-1 [44] suggests.

A.4.3 Method for time-referenced skip/seek

Starting from the original `skip` and `seek` methods, but adding functionality for traversing the transport stream for a PCR timestamp in the appropriate range. The first step is to search for the nearest PCR, and from that, calculate the value to search for. Besides new GET parameters, this will require client side changes for reading the preferred time skip and calling the server.

A.4.4 Implementation on Hibox Systems' PVR server

In order to implement the functions, the Player servlet had to be modified slightly to function stand-alone. The servlet is developed on Apache Tomcat 5.5, and generally, VLC is used locally for testing. The commands for controlling the playback can be sent in a web browser, since the HTTP protocol is stateless and the origin of a request does not make a difference. The servlet keeps track of sessions internally and applies the commands on the appropriate video stream. The implementations have also been tested with Hibox Systems' server and the Kreatel IPTV set-top box.

The results are highly subjective; in no trivial way is it possible to technically measure the amount of block artefacts, although it is clearly visible if a method counteracts the problem. What also is relevant is if a method affects the delay or crash tendency of the Kreatel receiver.

A.4.4.1 Method 1: transport stream level

In order to synchronize the read position to a transport stream packet, the method `static void syncTS (RandomAccessFile TS)` was devised. It will ensure that the transport stream sync character exists 188 bytes backwards aswell as 188 bytes forward, since the

character may also exist in the payload. On the same time, the read buffer was changed from 4096 bytes to 3948 (21*188) in order to read entire TS packets.

As could be expected, this did not make a significant difference. Block artefacts are still present, and the set-top box still has a tendency to stop or crash.

A.4.4.2 Method 2: packetized elementary stream level

This extends method 1 by initially synchronizing to a transport stream packet. The `static void syncPayloadStart (RandomAccessFile TS, int PID)` will, after calling `syncTS`, start traversing the transport stream file packet by packet until a packet with the correct PID and the payload_start flag is found.

This method makes a significant improvement with regard to block artefacts. The set-top box, aswell as VLC, still has a tendency to stop, and sometimes completely crash. The set-top box has a logging facility that gives a "PTS out of sync" message. This would mean that it cannot properly handle the sudden discontinuity in the timing information.

A.4.4.3 Method 3: elementary stream level

This method in turn extends method 2; after finding a packet with payload_start, it will further examine its payload. The function `static boolean validStartPES (byte[] data)` returns true if the byte array data contains a stream-sequence-GOP header combination.

The function does not perform noticeable better than method 2, possibly even slightly worse. By saving the stream to a file, it can be confirmed to be the expected structures. The likely cause is the inconsistent PCR aswell as PTS/DTS information.

A.4.5 Time-referenced skip/seek

The Player servlet was extended with the GET parameters `skipms` and `seekms` with accompanying methods. The parameter `mseconds` was also added to transfer the intended number of milliseconds. This value may be negative for `skipms`. For example, this request will move the playback position backwards around five seconds:

```
http://servername/player/Player?session=1&recording=37996
↪ &action=skipms&mseconds=-5000
```

The `skipms` function will initially stop the playback, find the next PCR reference, create a `PCR` object and save the timing information. After that, it starts going through the file in the correct direction until a PCR reference with the calculated value is found. Finally, `syncPayloadStart` is called and the playback starts again. The `seekms` function is almost similiar, but starts from the beginning of the file.

The client software was changed to call the Player servlet with the `skipms` parameter and a time of 30 seconds. By playing a recording with a visible clock, it was confirmed that the function is down to one second correct.

A.4.6 The final implementation

During testing of the time referenced skip/seek it was observed that the MPEG decoder did not seem to stop as much as with the methods aimed at solving the block artefact tendencies. The difference in the time referenced skip/seek is that before `syncPayloadStart` is run, the read position of the `RandomAccessFile` object will be at a transport stream packet containing PCR information. This means the playback will begin from the first packet containing a new GOP after the PCR. To further investigate, the function `static void syncToPCR(RandomAccessFile TS, int PCR_PID)` was implemented. It will search for the nearest PCR information. This can be considered the best possible result without thorough real-time modifications of the transport stream. One explanation for this behavior could be that the MPEG decoders in VLC and the Kreatel set-top box tolerate inconsistencies in the timing information better the longer it has time to adapt until the next time stamp. The PTS and DTS information that is sent in the video stream header may help the receiver tune into synchronization.

A.5 Implementation, client side

The implementation is largely server based but some changes need to be made to the middleware software, which makes up the user interface on the IPTV receiver. Primarily, functions have to be added to call the time referenced skip/seek. This system is based on HTML and JavaScript, and generally developed by the operator. It can be adapted to customer specifications, and is downloaded by the set-top box for a common HTTP server running for example Apache HTTP Server or Apache Tomcat.

For the implementation, Kreatels/Motorolas IPTV set-top box IP-STB 1510 is used. According to the specifications [62] it has a 420 DMIPS processor and 128 megabytes of memory. Only input is an RJ45 connector for 10/100 Mbps Ethernet. Outputs are SCART, composite video, s-video, analog audio and digital 5.1 surround. It also has a smart card reader for Conax based channel cards. The receiver is Linux based and does not have a non-volatile memory. It loads its system image over the network, from a special server that continuously sends out system images as UDP multicast. The programming environment consists of HTML and JavaScript to build up the user interface. In order to send HTTP requests to the server for controlling playback, the AJAX technique is used. It allows

sending an asynchronous request without loading a URL in the internal browser.
One example of AJAX-style programming on the set-top box:

```
function pauseServer() {
    var httpRequest = new XMLHttpRequest();
    httpRequest.onreadystatechange = function(){
    };
    httpRequest.open('GET',document.streams.sel.value+'&action=pause-resume',true);
    httpRequest.send(null);
    if (serverplaying == 0) serverplaying = 1;
    else serverplaying = 0;
}
```

This function is called when the Play button is pressed while playing, indicating a request for pause of playback. In the receiver, each remote control button can be associated with a function.

An example portal was developed to demonstrate how the functions can be utilized. This is largely based on Kreatels examples and Hibox Systems middleware. The most important part is reading the number of seconds to skip in the stream. This is handled by starting a timer when the user presses fast forward/backward on the remote control. If more key presses are received, the skip time is incremented/decremented and the timer reset.

When jumping forward, it is only necessary to send an AJAX request with to the server with the wanted `skipms` parameter. However, jumping backwards in timing information is forbidden in ISO/IEC 13818-1 [44] and it also has a tendency to cause the receiver to crash. The solution is to stop the playback and open a new stream from the server. Since the servlet sends the transport stream file position in response to any request, this can be used with the `position` parameter when opening the new connection. It will point to a transport stream packet containing a new GOP following a PCR time stamp. This enables an instant and flawless transition.

A.6 Conclusion

The primary goal of my thesis project was to investigate, and if possible, solve the problem with block artefacts after skips (real-time splicing) in MPEG transport streams. Secondary was to implement a method for skipping a specified number of milliseconds, instead of bytes.

The theoretical research showed that a real-time splicing was technically possible, by thoroughly analyzing the MPEG structures. Due to the complexity of the MPEG format, systematic top-down approach through the different layers was neccessary. By analyzing

the structures, a usable algorithm could be devised. The method is not computationally demanding, neither does it require much memory. It does, however, cause disk activity, which may be an issue if many simultaneous clients use the functions.

The time referenced skip/seek works very well — it is correct down to one second. This in particular may cause disk activity issues, thus a possible improvement on the server side would be to estimate an initial jump. It is not efficient to search through the entire file. When moving backwards it is necessary to stop and restart the media player from client side, but it may be a good idea when jumping forwards aswell. Otherwise, the buffer memory of the set-top box causes a two second delay.

One problem when not restarting the media player is that it tends to lock up for a second after the skip. This is due to discontinuities in the timing information, causing the MPEG decoder to show still pictures until it is synchronized to the new timing information. By restarting the media player, an acceptable start is acquired. It would be possible to create a real-time seamless splicing, but this would impose generating new time stamps for the material.

Even though the real-time splicing turned out to be more complicated than I initially thought, it is still possible to avoid some of the problems. Generally, I have reached the goals of my thesis project and accomplished a realistic solution that can be used on a production system.

Bilaga B

Ordlista / förkortningar

AJAX

Asynchronous JavaScript and XML. Teknik vid webprogrammering för att överföra och presentera information utan att ladda om websidan.

API

Application Programming Interface, programmeringsgränssnitt till t.ex. ett systembibliotek som definierar hur bibliotekets funktioner skall användas.

ATSC

Advanced Television Systems Committee, amerikansk arbetsgrupp som hade till uppgift att utveckla ett digital TV-system. Det resulterande systemet kallas också ATSC.

BBS

Bulletin Board System, modemuppringt serversystem för diskussioner och filutbyte.

CIF

Common Intermediate Format, standardiserad upplösning av 352x288 bildpunkter med 25 bilder per sekund, respektive 352x240 punkter med 30 bilder per sekund.

CSS

Cascading Style Sheets, metod för att beskriva hur element skall placeras och presenteras på websidor.

DCT

Discrete Cosine Transform, matematisk metod för att dela upp en signal i frekvenskomponenter.

DOM

Document Object Model, standard för hur XML-dokument skall ritas upp och hur elementen kan refereras.

DSL

Digital Subscriber Line, digital dataöverföring på telefonlinjer.

DTS

Decode Time Stamp, del av synkroniseringsinformationen på den paketerade elementära videoströmmens nivå som anger när en ruta skall avkodas.

DVB

Digital Video Broadcasting, ursprungligen ett europeiskt samarbete som utvecklats till standarder för digitala TV-sändningar.

DVB-C

Digital Video Broadcasting - Cable, standard för digital-TV i kabel-TV-nät.

DVB-S

Digital Video Broadcasting - Satellite, standard för satellitsänd digital-TV.

DVB-T

Digital Video Broadcasting - Terrestrial, standard för marksänd digital-TV.

EPG, Electronic Program Guide

Programtablå som är tillgänglig på TV-skärmen — t.ex. genom text-TV eller användargränssnittet i en set-top-box.

ES, Elementary Stream

Bitström (ISO/IEC 13818-2/13818-3) av komprimerad video- eller audiodata som erhålls ur MPEG-enkoder.

FCC

Federal Communication Commission, amerikanska telekommunikationsmyndigheten.

HDTV

High Definition TeleVision, TV-sändningar med signifikant högre upplösning än standard.

HTTP

HyperText Transfer Protocol. Nätverksprotokoll på applikationsnivå i OSI-modellen, används för att överföra material bl.a. på World Wide Web.

Huffman-kodning

Förlustfri komprimering som ersätter ofta förekommande bitsekvenser med kortare symboler.

IANA

Internet Assigned Numbers Authority, huvudsakligen amerikanskt styrd organisatör som koordinerar bl.a. tilldelning av IP-adresser och teckenenkodning i nätverksprotokoll.

IETF

Internet Engineering Task Force, samling av arbetsgrupper som utvecklar protokoll och standarder för Internet.

IGMP

Internet Group Management Protocol. Används i IP Multicasting bl.a. för att ansluta sig till och lämna Multicast-grupper.

IP

Internet Protocol, protokoll på OSI-modellens nätverksnivå som sköter om addressering och leverans av data på Internet.

IPTV

TV-sändningar över IP-nätverk.

MBONE

Multicast backbone, experimentellt multicastnätverk.

MHP

Multimedia Home Platform, DVB:s öppna standard för interaktivitet. Baserad på Java.

Middleware

Mellanliggande programvara som ansluter olika komponenter. Exempelvis en portalmjukvara som sköter om kommunikationen mellan IPTV-terminal och externa system som VOD-server.

Multicast

Nätverksprotokoll som sänds samtidigt till flera mottagare.

MUSE

MUltiple Subsampling Encoding, äldre japanskt system för analog HDTV.

NAT

Network Address Translation, metod som används för att dölja många interna IP-addresser bakom en som är synlig utåt.

VHF

Very High Frequency, frekvensbandet 30-300 MHz.

MPEG

Moving Picture Experts Group, samarbete för standardisering av videokomprimering.

OCAP

OpenCable Application Platform, amerikansk standard för interaktiv TV. Bygger på MHP.

Oktav

Fördubbling eller halvering av frekvens.

OSI-modellen

Open Systems Interconnection. Modell för datakommunikation i sju lager. Lagren är nedifrån: fysisk, datalänk, nätverk, transport, session, presentation och applikationsnivå.

PCM

Pulse Code Modulation. Signal digitaliserad vid förbestämda tidspositioner, elementär typ av sampling.

PCR

Program Clock Reference. Tidsreferens i transportströmmen, som är en momentan sampling av enkoderns interna klocka.

PES, Packetized Elementary Stream

Paketerad elementär MPEG-bitström (ISO/IEC 13818-1).

PS, Program Stream

Programström, flera multiplexade MPEG-strömmar (ISO/IEC 13818-1), i allmänhet både video och audio. Används för ändamål med få parallella strömmar och välkänt felfritt medium — bl.a. DVD.

PTS

Presentation Time Stamp, del av synkroniseringsinformationen på den paketerade elementära video- och audioströmmens nivå som anger när informationen skall visas, i förhållande till PCR.

PVR

Personal Video Recorder, funktionalitet för att spela in önskade sändningar.

QoS, Quality of Service

Nätverksmekanismer för att garantera att material levereras till mottagaren i tid och utan störningar, exempelvis genom att ge högre prioritet åt viktig trafik.

RTP

Real-time Transport Protocol, UDP-baserat nätverksprotokoll på applikationsnivå i OSI-modellen, används för enklare strömmande media.

RTSP

Real Time Streaming Protocol, utökning av RTP som tillåter kontroll av videoströmmen såsom paus och snabbspolning.

SDTV

Standard Definition TeleVision. Normala upplösningen för TV-sändningar, i DVB 720x576 punkter.

TCP

Transmission Control Protocol, anslutningsorienterat nätverksprotokoll i OSI-modellens transportnivå. TCP tillhandahåller en garanterad dataöverföring med paket i rätt ordning, och sänder borttappade paket igen.

TS, Transport Steam

Transportström, system definierat av MPEG (ISO/IEC 13818-1) för att parallellt sända många video- och audioströmmar, ofta över störningsbenägna medium. Används i DVB.

UDP

User Datagram Protocol, anslutningslöst nätverksprotokoll i OSI-modellens transportnivå. Används ofta för kort och icke-kritisk kommunikation, eftersom UDP, till skillnad från TCP, saknar all form av leveransbekräftelse.

UHF

Ultra High Frequency, frekvensbandet 300-3000 MHz.

Unicast

Nätverksprotokoll som är addresserat endast åt en mottagare.

VOD, Video On Demand

System för att på kundens begäran sända ut, ofta avgiftsbelagt, videomaterial.

Litteraturförteckning

[1] Mats Röjne. *Digital-TV via mark, satellit och kabel*, kapitel Analog TV, ss 11–34. Studentlitteratur, 2004.

[2] Mats Röjne. *Digital-TV via mark, satellit och kabel*, kapitel Marksänd analog TV, ss 93–106. Studentlitteratur, 2004.

[3] Digita. Digital-tv:s utveckling i Finland. `http://www.digitv.fi/sivu.asp?path=7;1251;1276`, november 2006.

[4] Ulrich Reimers. *Digital Video Broadcasting (DVB); the international standard for digital television*, kapitel Digital Television - First Summary, ss 1–16. Springer-Verlag Berlin Heidelberg New York, 2001.

[5] DVB Project. `http://www.dvb.org/products_registration/`, november 2006.

[6] Steven Morris och Anthony Smith-Chaigneau. *Interactive TV Standards: A Guide to MHP, OCAP and JavaTV*, kapitel The Middleware Market, ss 1–19. Focal Press, 2005.

[7] Steven Morris och Anthony Smith-Chaigneau. *Interactive TV Standards: A Guide to MHP, OCAP and JavaTV*, kapitel Middleware Architecture, ss 41–60. Focal Press, 2005.

[8] MHP. MHP Update. `http://www.mhp.org/about_mhp/who_is_using_mhp/`, september 2006.

[9] Wikipedia. Artikel om radio på Internet. `http://en.wikipedia.org/wiki/Internet_radio`, november 2006.

[10] Wikipedia. Artikel om Internet-TV. `http://en.wikipedia.org/wiki/Internet_television`, november 2006.

[11] Wikipedia. Artikel om TVUnetworks. `http://en.wikipedia.org/wiki/TVUnetworks`, november 2006.

[12] BusinessWeek Online Steve Rosenbush. Kazaa, Skype, and now “The Venice Project”. http://www.businessweek.com/bwdaily/dnflash/content/jul2006/db20060724_713810.htm, juli 2006.

[13] Broadband Services Forum. IPTV Explained. http://www.broadbandservicesforum.org/White+Papers.17.lasso, november 2006.

[14] Broadband Services Forum. The IPTV Bundle. http://www.broadbandservicesforum.org/White+Papers.17.lasso, november 2006.

[15] Broadband Services Forum. IPTV Versus Internet TV. http://www.broadbandservicesforum.org/White+Papers.17.lasso, november 2006.

[16] Ralph Wittmann och Martina Zitterbart. *Multicast Comunication Protocols and Applications*, kapitel The Basics of Group Communication, ss 16–20. Morgan Kaufmann, 2000.

[17] Juan-Mariano de Goyeneche. Multicast over TCP/IP HOWTO. http://tldp.org/HOWTO/Multicast-HOWTO.html, mars 1998.

[18] Wikipedia. Artikel om Video On Demand. http://en.wikipedia.org/wiki/Video_on_demand, november 2006.

[19] Tino Pyssysalo och Leo Ojala. *A High-Level Net Model of a Video on Demand System*, kapitel Introduction, ss 1–3. Helsinki University of Technology, Department of Computer Science, 1995.

[20] Wikipedia. Artikel om Kreatel. http://en.wikipedia.org/wiki/Kreatel, november 2006.

[21] Jari Aho. The Base Box Environment - Description and Draft Specification. http://www.acreo.se/upload/Lab/Testbed/DraftBaseboxenvironment2.0.1.pdf, mars 2006.

[22] Moving Picture Experts Group. About MPEG. http://www.chiariglione.org/mpeg/about_mpeg.htm, november 2006.

[23] John Watkinson. *The MPEG Handbook*, kapitel Introduction to compression, ss 19–25. Focal Press, 2001.

[24] Wikipedia. Artikel om MPEG-1. http://en.wikipedia.org/wiki/MPEG1, november 2006.

[25] John Watkinson. *The MPEG Handbook*, kapitel Introduction to compression, ss 1–3. Focal Press, 2001.

[26] John Watkinson. *The MPEG Handbook*, kapitel MPEG video compression, ss 224–243. Focal Press, 2001.

[27] John Watkinson. *The MPEG Handbook*, kapitel MPEG video compression, ss 253–261. Focal Press, 2001.

[28] John Watkinson. *The MPEG Handbook*, kapitel MPEG video compression, ss 245–248. Focal Press, 2001.

[29] Ulrich Reimers. *Digital Video Broadcasting (DVB); the international standard for digital television*, kapitel JPEG and MPEG Source Coding of Video Signals, ss 59–64. Springer-Verlag Berlin Heidelberg New York, 2001.

[30] John Watkinson. *The MPEG Handbook*, kapitel MPEG video compression, ss 243–265. Focal Press, 2001.

[31] Mats Röjne. *Digital-TV via mark, satellit och kabel*, kapitel Digital bildkomprimering, s 67. Studentlitteratur, 2004.

[32] John Watkinson. *The MPEG Handbook*, kapitel Audio compression, ss 178–184. Focal Press, 2001.

[33] John Watkinson. *The MPEG Handbook*, kapitel Audio compression, ss 194–216. Focal Press, 2001.

[34] European Telecommunications Standards Institute. *Final draft ETSI EN 300 468 v1.7.1 (2005-12)*, kapitel Descriptors, s 36. 2005.

[35] John Watkinson. *The MPEG Handbook*, kapitel Program and transport streams, ss 329–339. Focal Press, 2001.

[36] International Organization for Standardization. *ISO/IEC 13818-2: 1995 (E)*, kapitel Video bitstream syntax and semantics, s 29. ISO/IEC, 1995.

[37] Mats Röjne. *Digital-TV via mark, satellit och kabel*, kapitel Transportströmmen, ss 79–92. Studentlitteratur, 2004.

[38] International Organization for Standardization. *ISO/IEC 13818-1: 2000(E)*, kapitel PES packet, ss 31–40. ISO/IEC, 2000.

[39] Digita. Kanaler. `http://www.digitv.fi/sivu.asp?path=7;1248`, december 2006.

[40] International Organization for Standardization. *ISO/IEC 13818-1: 2000(E)*, kapitel Specification of the Transport Stream syntax and semantics, ss 18–30. ISO/IEC, 2000.

[41] International Organization for Standardization. *ISO/IEC 13818-1: 2000(E)*, kapitel Program Stream bitstream requirements, ss 50–58. ISO/IEC, 2000.

[42] International Organization for Standardization. *ISO/IEC 13818-1: 2000(E)*, kapitel Program specific information, ss 41–50. ISO/IEC, 2000.

[43] Mats Röjne. *Digital-TV via mark, satellit och kabel*, kapitel Mottagare för DVB-T: CA-modulen, ss 170–172. Studentlitteratur, 2004.

[44] International Organization for Standardization. *ISO/IEC 13818-1: 2000(E)*, kapitel Annex K: Splicing Transport Streams, ss 137–139. ISO/IEC, 2000.

[45] Chris Shiflett. *HTTP Developer's Handbook*, kapitel Introducing HTTP, ss 15–20. Sams Publishing, 2003.

[46] Chris Shiflett. *HTTP Developer's Handbook*, kapitel The Internet and the World Wide Web, ss 21–28. Sams Publishing, 2003.

[47] Chris Shiflett. *HTTP Developer's Handbook*, kapitel HTTP Requests, ss 43–68. Sams Publishing, 2003.

[48] Chris Shiflett. *HTTP Developer's Handbook*, kapitel HTTP Responses, ss 69–90. Sams Publishing, 2003.

[49] Chris Shiflett. *HTTP Developer's Handbook*, kapitel General Headers, Entity Headers, ss 91–109. Sams Publishing, 2003.

[50] Chris Shiflett. *HTTP Developer's Handbook*, kapitel Maintaining State, ss 121–143. Sams Publishing, 2003.

[51] Wikipedia. Comparison of web servers. `http://en.wikipedia.org/wiki/Comparison_of_web_servers`, Januari 2007.

[52] The Apache Software Foundation. About the Apache HTTP Server Project. `http://httpd.apache.org/ABOUT_APACHE.html`, Januari 2007.

[53] Wikipedia. Artikel om Internet Information Services. `http://en.wikipedia.org/wiki/Internet_Information_Services`, Januari 2007.

[54] Wikipedia. Artikel om Java Servlets. `http://en.wikipedia.org/wiki/Java_Servlet`, Januari 2007.

[55] Wikipedia. Artikel om Apache Tomcat. http://en.wikipedia.org/wiki/Apache_Tomcat, Januari 2007.

[56] Colin Perkins. *RTP: Audio and Video for the Internet*, kapitel An introduction to RTP, ss 3–14. Addison-Wesley Professional, 2003.

[57] Colin Perkins. *RTP: Audio and Video for the Internet*, kapitel The Real-time Transport Protocol, ss 51–66. Addison-Wesley Professional, 2003.

[58] The Internet Society. RFC 2326 Real Time Streaming Protocol (RTSP). http://tools.ietf.org/html/rfc2326, April 1998.

[59] myIPTV. Quick and dirty overview of the RTSP protocol. http://www.myiptv.org/Articles/tabid/129/ctl/Details/mid/508/ItemID/0/Default.aspx, Januari 2006.

[60] Colin Perkins. *RTP: Audio and Video for the Internet*, kapitel RTP Control Protocol, ss 95–128. Addison-Wesley Professional, 2003.

[61] Faultline. Motorola makes up for lost time with Kreatel buy. http://www.theregister.co.uk/2006/01/20/motorola_iptv_kreatel, Januari 2006.

[62] Motorola. Motorola VIP1500 Series MPEG-2 IP Set-Top Series. http://www.motorola.com/mot/doc/6/6343_MotDoc.pdf, 2006.

[63] Wikipedia. Artikel om AJAX. http://en.wikipedia.org/wiki/AJAX, April 2007.