UNIVERZA V LJUBLJANI
FAKULTETA ZA RAČUNALNIŠTVO IN INFORMATIKO

Anže Rehar

Implementacija protokola HTTP Live Streaming v programskem jeziku Java

DIPLOMSKO DELO
NA UNIVERZITETNEM ŠTUDIJU

Mentor: doc. dr. Matija Marolt

Ljubljana, 2012

Besedilo je oblikovano z urejevalnikom besedil LaTeX.

Univerza *v Ljubljani*
Fakulteta *za računalništvo in informatiko*

Tržaška 25
1001 Ljubljana, Slovenija
telefon: 01 476 84 11
01 476 83 87
faks: 01 426 46 47
01 476 87 11
www.fri.uni-lj.si
e-mail: dekanat@fri.uni-lj.si

Št. naloge: 01840/2012
Datum: 02.04.2012

Univerza v Ljubljani, Fakulteta za računalništvo in informatiko izdaja naslednjo nalogo:

Kandidat: **ANŽE REHAR**

Naslov: **IMPLEMENTACIJA PROTOKOLA HTTP LIVE STREAMING V PROGRAMSKEM JEZIKU JAVA**

IMPLEMENTATION OF HTTP LIVE STREAMING PROTOCOL IN JAVA

Vrsta naloge: Diplomsko delo univerzitetnega študija

Tematika naloge:

V diplomskem delu opišite različne načine dostave video vsebin na mobilne naprave in spletne brskalnike. Osredotočite se na protokole, ki med delovanjem prilagajajo pasovno širino in podrobno opišite protokol HTTP Live Streaming. Prikažite prednosti in slabosti protokola ter ga primerjajte z glavnimi konkurenti. Implementirajte protokol v programskem jeziku Java.

Mentor:

doc. dr. Matija Marolt

Dekan:

prof. dr. Nikolaj Zimic

IZJAVA O AVTORSTVU

diplomskega dela

Spodaj podpisani Anže Rehar,

z vpisno številko 63050096,

sem avtor diplomskega dela z naslovom:

Implementacija protokola HTTP Live Streaming v programskem jeziku Java

S svojim podpisom zagotavljam, da:

- sem diplomsko delo izdelal samostojno pod mentorstvom
  doc. dr. Matije Marolta
- so elektronska oblika diplomskega dela, naslov (slov., angl.), povzetek (slov., angl.) ter ključne besede (slov., angl.) identični s tiskano obliko diplomskega dela
- soglašam z javno objavo elektronske oblike diplomskega dela v zbirki "Dela FRI".

V Ljubljani, dne 15.10.2012 Podpis avtorja:

Zahvala

Zahvaljujem se doc. dr. Matiji Maroltu za mentorstvo pri diplomski nalogi. Hvala za razpoložljivost in prijaznost. Posebna zahvala gre Metki za spodbujanje delovnega elana, nenehno podporo in ljubezni poln dom. Zahvala gre tudi staršem, ki so me v vseh letih študija podpirali in mi stali ob strani. Mihi Martincu in ostalim sodelavcem v podjetju Fora d.o.o. se zahvaljujem za prijetno delovno okolje in možnost dela na zanimivih projektih.

Kazalo

Seznam uporabljenih kratic in simbolov

3GPP - 3rd Generation Partnership Project
AAC-LC - Advanced Audio Coding - Low-Complexity
ADT - Android developer tools
AES - Advanced Encryption Standard
API - Application programming interface
ATM - Asynchronous Transfer Mode
ATSC - Advanced Television Systems Committee
BPMN - Business Process Modeling Notation
EDGE - Enhanced Data rates for GSM Evolution
CDN - Content Delivery Network
CPE - Centralna procesna enota
DASH - Dynamic Adaptive Streaming over HTTP
DVB - Digital Video Broadcasting
HD - High definition
HDS - HTTP Dynamic Streaming
HLS - HTTP Live Streaming
HTTP - HyperText Transfer Protocol
IDC - International Data Corporation
IDE - Integrated development environment
IEC - International Electrotechnical Commission
IETF - Internet Engineering Task Force
IP - Internet protocol
IPTV - Internet Protocol television
ISO - International Organization for Standardization
J2ME - Java 2 Platform, Micro Edition
MIME - Multipurpose Internet Mail Extensions
MJPEG - Motion JPEG
MPEG - Moving Picture Experts Group

MSS - Microsoft Smooth Streaming
OSMF - Open Source Media Framework
PIFF - Protected Interoperable File Format
RFC - Request for Comments
RTMP - Real Time Messaging Protocol
RTP - Real-time Transport Protocol
RTSP - Real Time Streaming Protocol
SD - Standard definition
SDK - Software development kit
UDP - User Datagram Protocol
UML - Unified Modeling Language
URI - Uniform Resource Identifier
UTF - Unicode Transformation Format
W3C - World Wide Web Consortium
TCP - Transmission Control Protocol
TS - MPEG transport stream

Povzetek

V diplomskem delu se ukvarjamo z dostavo video vsebin na mobilne naprave. Visoka rast prodaje pametnih telefonov in tablic v zadnjih letih sili založnike k iskanju novih možnosti za dostavo vsebin uporabnikom. Ker mobilne naprave prinašajo določene omejitve, postane implementacija pretakanja videa na te naprave zapleten projekt. Največji vpliv na projekt ima izbira protokola za pretakanje. Kljub temu da še ni splošno sprejetega standarda, je na voljo množica tehnologij za pretakanje videa, ki delujejo preko protokola HTTP. V diplomskem delu opišemo različne načine dostave video vsebin na mobilne naprave in spletne brskalnike. Zaradi omejitev v hitrosti povezave se osredotočimo na protokole, ki med delovanje prilagajajo pasovno širino. V osrednjem delu diplomskega dela podrobno opišemo izbrani protokol HTTP Live Streaming. Opišemo postopek od priprave vsebine in segmentacije do izdelave seznamov predvajanja in prenosa na naprave uporabnikov. Prikažemo prednosti in slabosti protokola HLS ter ga primerjamo z glavnima konkurentoma - Microsoftovim Smooth Streaming in protokolom HTTP Dynamic Streaming podjetja Adobe. Glavni razlog teme te diplomske naloge pa je implementacija protokola HLS v programskem jeziku Java. Razvita je bila programska knjižnjica, ki se lahko uporabi na poljubnih operacijskih sistemih in platformah. V zadnjem delu je tako opisana arhitektura in delovanje ter glavne težave, s katerimi smo se srečali pri razvoju. Kot dokaz o uspešni implementaciji protokola prikažemo uporabo programske knjižnjice v dveh aplikacijah.

Ključne besede:

HTTP Live Streaming, video pretakanje, prilagodljiva pasovna širina, algoritmi za prilagajanje pasovne širine, HTTP Dynamic Streaming, Microsoft Smooth Streaming, kodiranje videa, Android

Abstract

In this thesis we discuss protocols for streaming to mobile devices. Tremendous growth of smartphone and tablet sales numbers over the past few years push content providers to seek new ways to offer video content. However, these devices have some limitations that make the implementation of video streaming a complex project. Most depends on the streaming protocol selection. There is a set of HTTP based technologies but there are no standards for adaptive streaming to mobile devices. In thesis we describe different ways of streaming video content to mobile devices and browsers. Due to network speed limitations we are focusing on protocols that have mechanisms for adaptive bitrate streaming. In main part of the thesis we describe HTTP Live Streaming in detail. We describe a process of preparing video content and the segmentation process of video stream. Next step is preparation of playlists and streaming to mobile devices. We show the strengths and weaknesses of HLS protocol and compare it to its main competitors - Microsoft Smooth Streaming and Adobe HTTP Dynamic Streaming. Main motivation and reason for this thesis is the development of program library in Java programming language that implements HLS protocol. Library can be used on various operating systems and platforms. In last part we describe the architecture and process workflow together with main issues we encountered during the development. As an evidence of successful implementation two applications are shown that both use this software library.

Key words:

HTTP Live Streaming, video streaming, adaptive streaming, rate-adaptation algorithms, HTTP Dynamic Streaming, Microsoft Smooth Streaming, video encoding, Android

Poglavje 1

Uvod

V času, ko proizvajalci pametnih telefonov, tablic in drugih prenosnih naprav vedno bolj vključujejo brezžično povezljivost in lastniki vsebin iščejo nove možnosti neposredne dostave teh vsebin uporabnikom, so vse bolj v porastu tudi aplikacije za prenos multimedijskih vsebin na mobilne naprave.

1.1 Opis problema

Pri dostavi video vsebin se ponudniki srečujejo z določenimi problemi, ki vplivajo na razvoj in implementacijo rešitev. Mobilni operaterji zapirajo določena vrata protokola TCP/IP na mobilnih omrežjih in tako onemogočajo uporabo protokolov, ki so široko razširjeni pri klasičnih sistemih kot je na primer IPTV, kjer operaterji storitve ponujajo samo znotraj svojih lastnih omrežij. Naslednji problem pri pretakanju videa preko mobilnih omrežij do naprave uporabnika je preklapljanje med baznimi postajami mobilnih omrežij ali med dostopnimi točkami Wi-Fi. Ker se lahko med preklopom pojavi začasna prekinitev v omrežni povezljivosti mora rešitev poskrbeti za predpomnenje tudi v primeru predvajanja žive slike. Hkrati je potrebno stremeti k čim manjši porabi podatkovnega prenosa pri povezovanju preko mobilnih omrežij.

Pri fiksnih internetnih priključkih gospodinjstev se prav tako pojavljajo določene omejitve zaradi prevajanja omrežnega naslova kot tudi zaradi zaprtih vrat na domačih usmerjevalnikih. Dodatne nastavitve lahko sicer odpravijo omejitve a zahtevajo od uporabnikov dodatno znanje in zmanjšujejo varnost domačega omrežja. Navkljub temu, da je povprečna hitrost dostopa do interneta gospodinjskih odjemalcev stalno v porastu, lahko ta pri nekaterih še vedno predstavlja omejitev ob hkratni uporabi več storitev, ki se med seboj

borijo za zadosten kos pasovne širine. Zaradi tega je potrebno najdi način, ki uporabnikom omogoča spremljanje videa pri optimalnem izkoristku hitrosti omrežne povezave.

Širše gledano je vsem problemom in zahtevam skupno to, da mora rešitev zagotavljati več različinih kvalitetnih nivojev podatkovih tokov in omogočati neprekinjeno predvajanje ob trenutkih omejene internetne povezave. Ob tem pa mora vse to za uporabnika potekati neopazno in samodejno.

Pri izbiri protokola, ki zadošča zgornjim zahtevam pa lahko naletimo na nepodporo protokola za sistem za katerega želimo razviti aplikacijo ali pa podpora ni na voljo na starejših napravah in nas to omejuje, če želimo storitve ponuditi kar najširšemu krogu uporabnikov.

1.2 Motivacija za delo

Odločitev za diplomsko nalogo s tega področja je posledica večletnega dela v podjetju Fora d.o.o., kjer v sklopu IPTV platforme razvijamo tudi aplikacije za mobilne naprave (J2ME, iOS, Android). V zadnjem obdobju je bila razvita aplikacije za mobilno televizijo na operacijskih sistemih Android, iOS, Windows, Mac OS X in Linux za slovenskega telekomunikacijskega operaterja T-2. Ob odločitvi za uporabo protokola HTTP Live Streaming se je pojavila potreba po lastni implementaciji. Vzrok je bil v nepodpori tega protokola na starejših napravah in samo delni podpori na novejših napravah Android. Na operacijskih sistemih za osebne računalnike pa te podpore s strani sistema sploh ni bilo. Poleg navedenega pa se z lastno implementacijo pridobi tudi možnosti nadzora ob morebitnih spremembah ali dopolnitvah protokola.

Ob pregledovanju domače literature, ki zadeva opisano področje smo ugotovili, da je le to slabo pokrito. Za protokol HLS, ki je glavni del te diplomske naloge, v slovenščini še ni možno najti literature ali zaslediti omembe tega protokola kljub temu, da ga v praksi uporablja že več slovenskih podjetij. To je po eni strani tudi razumljivo, saj je protokol še mlad in v fazi hitrega razvoja. Tako je motivacija za diplomsko nalogo tudi priprava splošnega pregleda znanja o področju, ki je precej kompleksno ter razdrobljeno in je lahko v pomoč morebitnim drugim razvijalcem podobnih rešitev in drugim, ki jih to področje zanima.

1.3 Rešitev problema

Probleme iz drugega dela tega poglavja smo rešili z uporabo protokola HTTP Live Streaming. Rešitev na strani odjemalcev predstavlja lastna programska knjižnjica za protokol HLS. Pri HLS se ves promet odvija izključno preko protokola HTTP. Knjižnjica je napisana v programskem jeziku Java in omogoča uporabo na različnih operacijskih sistemih za osebne računalnike kot tudi za operacijski sistem Android, ki skupaj z operacijskim sistemom iOS predstavlja večinski delež trga pametnih mobilnih telefonov, tablic in drugih prenosnih naprav.

1.4 Zgradba diplomskega dela

V drugem poglavju se opiše okolje in na splošno obravnava prenos videa na mobilne naprave. Tu so navedene različne možnosti prenosa videa preko omrežij na naprave. En razdelek je namenjen opisu pretakanja s prilagodljivo pasovno širino.

Tretje poglavje se nanaša na protokol HLS. V začetku je podrobno opisano delovanje in celoten postopek od priprave vsebine pri ponudniku do predvajanja na napravah uporabnikov. V nadaljevanju so naštete prednosti in slabosti izbranega protokola ter primerjava z drugimi protokoli s prilagajanjem pasovne širine. Predvsem z glavnima konkurentoma, ki sta Micsrosoft Smooth Streaming in Adobe HTTP Dynamic Streaming.

V četrtem poglavju je opisan praktičen primer implementacije protokola HLS v programskem jeziku Java, ki se lahko uporabi tudi za operacijski sistem Android. Navedena so uporabljena orodja in zgradba knjižnjice ter opis delovanja. Ob koncu so prikazani zaslonski posnetki dveh aplikacij, kjer se knjižnjica že uporablja. Omenjeni so tudi problemi s katerimi smo se srečevali.

V zadnjem poglavju se povzame delo, ki je bilo opravljeno. Opredeli se tudi pomembnost diplomskega dela ter predlaga možne dopolnitve in izboljšave.

Poglavje 2

Pretakanje videa na prenosne naprave

2.1 Okolje

Ogled televizije in drugih video vsebin v spletnih brskalnikih in na mobilnih napravah je postalo nekaj običajnega in široko uporabljenega. Kot primer lahko podamo spletnega velikana YouTube ki navaja[8], da si preko njihovega servisa vsak mesec preko 800 milijonov uporabnikov ogleda za preko 3 milijarde ur video vsebin. V zadnjem času še posebno veliko rast dosega uporaba servisa preko prenosnih naprav. V letu 2011 se je te vrste promet povečal za trikrat in že dosega 10% vseh ogledov.

Vodilni svetovni ponudnik raziskav in analiz na področju globalne industrije informacijske tehnologije Gartner v poročilu[9] iz začetka leta navaja, da je bilo samo v letu 2011 prodanih 472 milionov pametnih telefonov, kar je 58% rast glede na leto 2010. Glede na tržni delež operacijskih sistemov za te naprave glavna konkurenta Applov iOS in Googlov Android obvladujeta skoraj tri četrtine trga. Razdrobljenost trga in potrebe po ločenem razvoju za vsako platformo posebej vplivajo na odločitve ponudnikov vsebin o tem katere platforme bodo podprli. S tem so ostale platforme posredno postavljene v še slabši položaj. V tem trenutku se večina ponudnikov odloča ravno za prej omenjeni platformi in s tem pospešuje nižanje tržnega deleža ostalih platform kot sta naprimer Symbian in BlackBerry OS.

Strojno dekodiranje kodeka H.264 je vključeno v večino novejših procesorjev za mobilne naprave. Dekodiranje WebM po drugi strani še ni strojno podprto, prav tako je programsko samo na nekaterih mobilnih napravah. Goo-

gle dela tudi na strojni implementaciji a je ta še daleč do splošne razširjenosti. Založniki se tako osredotočajo na kodek H.264, ki postaja defacto standard. Tudi Google kljub napovedim o ukinitvi podpore kodeku H.264 za HTML5 video v svojem brskalniku Chrome te podpore še ne ukinja.

Ker ponudniki vsebin nimajo vpliva na kvaliteto internetne povezave, lahko uporabniki izkusijo zatikajoče video predvajanje ali daljše premore med predvajanjem. Tudi če so uporabniki priklopljeni na hitre povezave, si jih dostikrat delijo z drugimi in tako ob gledanju doživijo slabšo uporabniško izkušnjo.

Uporabniki želijo možnost ogleda video vsebin na različnih napravah in ob času ko to sami želijo. Kot dodatne funkcionalnosti cenijo možnost zaustavitve predvajanja in nadaljevanja ogleda tudi v primeru predvajanja v živo. Predvajanje bi moralo potekati brez zatikanj ali prekinitev ob najvišji možni kvaliteti, ki jo dopušča uporabnikova internetna povezava.

2.2 Pregled protokolov za pretakanje

V preteklosti so se večinoma uporabljali protokoli kot so RTP skupaj z RTSP, dandanes pa se tehnologije za prenos opirajo predvsem na protokol HTTP in so hkrati načrtovane tako, da učinkovito delujejo v velikih razpršenih omrežjih kot je internet. Kljub navidezno pestri izbiri možnosti smo zaradi določenih omejitev na posameznih platformah dostikrat omejeni pri izbiri protokola.

2.2.1 Stalen pretok

Pri protokolih s pravim stalnim pretokom podatkov se avdio in video podatki pošiljajo do končnega uporabnika v bolj ali manj konstantem toku. To pomeni, da se uporabniku prenešeni podatki predvajajo sproti. Zaradi tega ti tradicionalni protokoli med strežnikom in odjemalcem ohranjajo stalno povezavo, ki omogoča komunikacijo med njima v realnem času. Za dostavo samih podatkov se uporabljajo namenski protokoli, ki so osnovani na protokolih TCP ali UDP.

Glavni problem takšnega pristopa je, da je pretakanje po teh protokolih dostikrat blokirano s strani požarnih zidov, ki ponavadi dopuščajo samo protokol HTTP na vratih 80. Da bi obšli ta problem so se za te protokole razvile razširitve, ki ovijejo tok podatkov v HTTP zahtevke. Ta tehnika se imenuje tuneliranje, ki pa zaradi dodatnih podatkov zmanjšuje kapaciteto in s tem zmogljivost. Zaradi tega se ta rešitev največkrat pojavlja samo kot nadomestna. Protokoli, ki spadajo v segment protokolov s stalnim pretokom so:

- Real Time Streaming Protocol (RTSP): razvit in predstavljen od IETF leta 1998 v obliki zahteve za mnenje - RFC 2326. Večina strežnikov RTSP za dostavo medijskega toka podatkov uporablja Real-time Transport Protocol (RTP), ki podpira širok spekter različnih formatov kot so H.264, MJPEG, MPEG-2 in drugih. Na strani odjemalcev ta protokol podpirajo različne aplikacije kot so QuickTime, VLC, Skype in Windows Media Player. Večina današnjih pametnih telefonov podpira RTSP kot del standarda 3GPP. Kot primer lahko navedemo platforme Android, Blackberry ter Symbian a ne iOS.

- Real Time Messaging Protocol (RTMP): razvilo ga je podjetje Adobe Systems in njegovo specifikacijo nedolgo nazaj ponudilo tudi v javno uporabo. RTMP je v osnovi uporabljen za pretakanje zvoka in videa preko interneta na odjemalce, ki imajo nameščen Adobe Flash Player. Zaradi velikega uspeha predvajalnika Flash v preteklosti se dandanes velik delež pretakanja preko interneta odvija ravno preko tega protokola ali njegovih različic (RTMPT, RTMPS, RTMPE ...).

- MPEG transport stream (MPEG-TS): je mehanizem, definiran v protokolu MPEG-2 protocol, ki kontrolira prenos in hrambo avdio in video podatkov. Največ se uporablja v prenosnih sistemih kot sta DVB in ATSC ter na mnogih konvencionalnih vmesnikih za IPTV.

Večina teh protokolov ponuja tudi varnostne funkcije, ki uporabljajo enkripcijske algoritme za zaščito podatkovnih tokov. Ker vsi ti protokoli potrebujejo stalno povezavo med strežnikom in odjemalcem so s stališča stroškov pri nakupu strežnikov lahko slabši.

2.2.2 HTTP progresivno pretakanje

HTTP progresivno pretakanje ali psevdopretakanje kot že ime nakazuje ni pravo pretakanje ampak oblika progresivnega prenašanja, ki uporabnikom omogoča iskanje ozirona premik na dele video toka, ki se še niso prenesli. Protokol za prenašanje je klasični HTTP, ki je združen z mehanizmom, ki predvajalniku omogoča, da v HTTP zahtevek vključi parameter s časovnim žigom. S pomočjo tega mehanizma se lahko odjemalec kadarkoli poljubno premakne po časovnem traku medijske vsebine.

V primerjavi s protokoli s stalnim pretokom ima prevdopretakanje nekaj prednosti:

- Povzroča manj sitnosti zaradi požarnih zidov in preusmerjevalnih strežnikov (predvsem znotraj omrežij velikih združb).
- Začetni čas nalaganja je ponavadi krajši.
- Omogoča boljšo uporabniško izkušnjo pri slabih omrežnih povezavah.

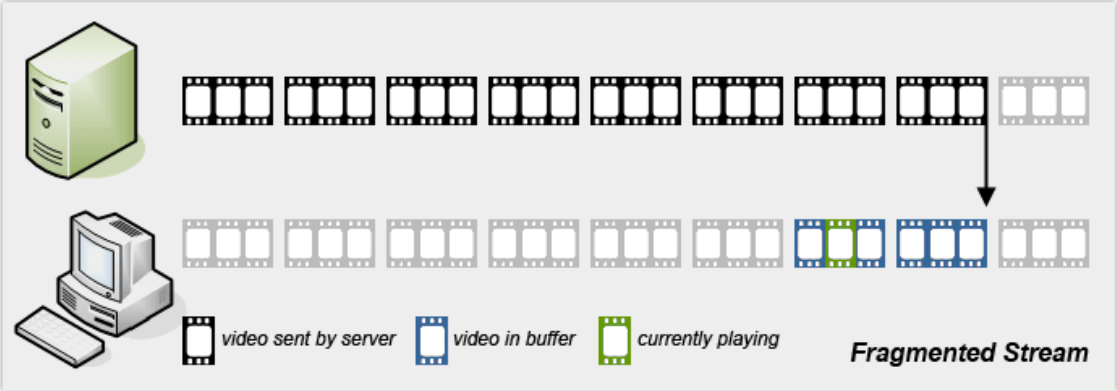
Prav tako pa ima progresivno pretakanje nekaj slabosti:

- Za razliko od protokolov s stalnim pretakanjem se tu prenešeni podatki prenesejo na fizični pogon naprave uporabnika.
- Povzroča večjo količino prenešenih podatkov, saj si protokoli prizadevajo prenesti celotne video datoteke. Na drugi strani protokoli s stalnim prenosom prenesejo samo podatke, ki si jih uporabnik dejansko ogleda.
- Pri iskanju po časovnem traku daljših video vsebin prevdopretakanje potrebuje relativno dolgo časovno zakasnitev preden se predvajanje lahko nadaljuje.
- Pri vsebinah z veliko pasovno širino lahko upočasni ali celo povzroči sesutje uporabnikovega brskalnika.
- Progresivno prenašanje ne omogoča tako natančnega beleženja statistike gledanja saj ni nujno, da si je uporabnik vse prenešene dele medijske vsebine dejansko ogledal.

2.2.3 HTTP pretakanje

Pri pretakanju videa preko protokola HTTP pravzaprav ne gre za čisto neprekinjeno pretakanje ampak za prenašanje posameznih delov video toka v ločenih kosih kot lahko vidimo na sliki 2.1. HTTP pretakanje je novejša tehnologija, ki se še razvija in bi veliko pridobila s sprejetjem splošnega standarda. Trenutno so na voljo ločene rešitve različnih proizvajalcev, ki v principu vse uporabljajo enake mehanizme, si pa med seboj niso kompatibilne. Za uporabo pa odjemalci potrebujejo prodajalčevo lastno programsko opremo ali platformo.

Prenašanje preko teh protokolov deluje tako, da se celoten tok podatkov razdeli na posamezne segmente oziroma datoteke, ki so primerne za enostaven prenos preko protokola HTTP. Zaradi uporabe tega protokola težav zaradi požarnih zidov načeloma ni.



Slika 2.1: Fragmentiran video tok

Naslednja prednost je možnost hranjenja segmentov v obliki datotek na strežnikih bližje odjemalcem ali pri ponudnikih omrežnih storitev. To pri ponudnikih vsebin zmanjša skupno porabo pasovne širine, ki je potrebna za dostavo vseh vsebin določenemu številu strank. To pri protokolih kot je RTMP ni bilo mogoče. V praksi se je ta prednost sicer izkazala kot zanemarljiva, o čimer govori raziskava[11] podjetja Streaming Media.

Slabost takšnega pristopa je manjša varnost zaščite avtorskih pravic. Zaradi odsotnosti kakršnihkoli metapodatkov pri prenosu je tudi težje zagotovoti kvaliteto storitev.

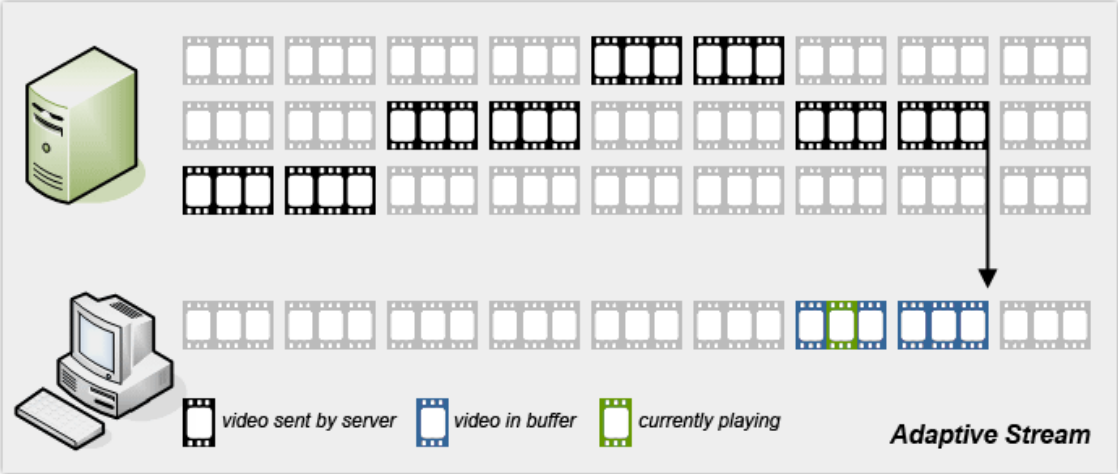
V želji po standardizaciji video pretakanja preko HTTP je Motion Pictures Experts Group (MPEG) ustvaril specifikacijo, ki se imenuje Dynamic Adaptive Streaming over HTTP (DASH) in jo novembra leta 2011 tudi objavil kot standard ISO/IEC 23009-1. V vsakem primeru se mora pred splošno uveljavitvijo standarda rešiti še veliko problemov, ki zadevajo tudi patentne spore in druga pravna vprašanja. Potem bo lahko DASH postal splošno uveljavljen standard od katerega bi pridobili tako ponudniki vsebin kot uporabniki. Kar pa najverjetneje ni za pričakovati saj so posamezna podjetja v zadnjih letih že razvila konkurenčne protokola in jih močno razširila tako med ponudniki vsebin kot na naprave uporabnikov. Vsi trije najbolj razširjeni protokoli, ki uporabljajo opisano tehniko so podrobneje predstavljeni v tretjem poglavju.

2.3 Pretakanje s prilagodljivo pasovno širino

Pričakovano je, da želijo uporabniki spremljati video ob najboljši možni kvaliteti a brez zakasnitev ob začetku predvajanja ali vmesnih prekinitev zaradi nalaganja, ki poslabšajo uporabniško izkušnjo. Da lahko internet postane glavni medij za dostavo video vsebin uporabnikom mora biti storitev zanesljiva, hkrati pa mora ponujati visoko kakovost vsebin. To zahteva visoko pasovno širino in nizek odzivni čas. V zadnjih letih se je kvaliteta slike ob prehodu na tehnologijo visoke ločljivost HD močno izboljšala. Prav tako so storitve dostopne na vedno več napravah, funkcionalnosti predvajalnikov pa so vedno bolj pestre. Vedno manj se tudi pojavlja potreba po namestitvah dodatnih programov ali vtičnikov za ogled video vsebin na določenem portalu. Pogosto posodabljanje programske opreme, nalaganje, medpomnenje in zatikanje videa kot tudi v nekaterih primerih še vedno slaba kvaliteta slike v primerjavi s klasično televizijo še vedno odvrnejo mnoge uporabnike od tega, da bi bilo predvajanje preko spleta ali preko mobilne naprave prva izbira za ogled video vsebin. Na drugi strani se pri satelitski in kabelski televiziji vedno bolj uveljavljajo programi v kvaliteti visoke ločljivosti HD, uporabniki pa tudi pri predvajanju preko interneta pričakujejo enako kvaliteto slike in zanesljivost storitev. Tem željam pa ni mogoče ugoditi zgolj z dvigom kvalitete predvajanja saj ta sprememba uporabniku prinaša prekinitve in druge nevšečnosti, ponudniku vsebin pa visoke stroške zaradi ogromnih zahtev po strojni opremi. Internetnim operaterjem pa visoke pasovne širine povzročajo obremenjenost mrežne infrastrukture.

Zaradi tega se vse več ponudnikov vsebin obrača na pretakanje s prilagodljivo pasovno širino (adaptive bitrate streaming). To je mehanizem, ki glede na pasovno širino in zmožnost procesiranja v realnem času izbira med različnimi pasovnimi širinami in s tem prilagaja kvaliteto video toka predvajanja. Za to je potrebno kodiranje enega video vira v več video tokov različnih pasovnih širin kot je prikazano na sliki 2.2. Odjemalec pa nato sproti samodejno izbira najvišji kvalitetni nivo, ki glede na trenutne razmere še omogoča gladko predvajanje.

Protokoli pri uporabniku zaznavajo omejitve v pasovni širini omrežne povezave in v zmogljivosti procesorskih enot. V realnem času nato glede na zbrane informacije prilagajajo kvaliteto video toka podatkov. Za to, da lahko takšen mehanizem deluje, je potreben kodirnik na strani strežnika, ki je sposoben kodiranja vhodnega multimedijskega toka v več različnih kvalitet v realnem času. Na strani odjemalca pa je potreben dekodirnik, ki je sposoben predvajanja video tokov različnih kvalitet in preklapljanja med njimi. Predvajalnik



Slika 2.2: Fragmentiran video tok z različnimi pasovnimi širinami

na odjemalcu tako ob predvajanju prenaša posamezne kose video toka in jih nazaj lepi v enoten video tok, ki pa lahko zaradi preskokov med kvalitetami vsebuje kose različnih kvalitet.

Ob dobri implementaciji to prinaša hiter čas vzpostavitve in začetka predvajanja, manjšo potrebo po predpomnenju in dobro uporabniško izkušnjo za različne kombinacije zmogljivosti naprav in različnih hitrosti omrežnih povezav. Neprekinjeno predvajanje je v večini primerov tekoče tudi pri večsekundni prekinitvi omrežne povezave ali med preklopom med dvema omrežnima povezavama. To se lahko zgodi ob preklopu med Wi-Fi in mobilnim podatkovim omrežjem ter med preklopi med posameznimi baznimi postajami mobilnih operaterjev. Za ugotavljanje izbire kvalitete predvajanja se uporabljajo različni algoritmi. Najbolj pogosto se uporabja čas prenosa predhodnih delov video toka. Glede na te podatke odjemalec izračuna, kateri kvalitetni nivo bo še lahko prenesel k sebi do trenutka, ko ga bo potreboval. Izbira kvalitete je tako popolnoma na strani odjemalca.

Poleg naštetega je seveda pomembno tudi dobro omrežje za dostavo teh vsebin pri ponudniku. Dodatna kvaliteta pri protokolu je možnost nastavljanja različnih pasovnih širin, velikosti okvirja videa in ostalih parametrov, da so možne sprotne prilagoditve, ko se na trgu pojavljajo nove naprave.

V primeru, da je rešitev kvalitetna, se končni uporabnik naj ne bi zavedal, da je video sestavljen iz večih tokov različnih pasovnih širin.

Kljub temu, da se pretakanje s prilagodljivo pasovno širino večinoma nanaša

Platrofma	Progressivno pretakanje	RTSP	RTMP	HLS	MSS	HDS
iOS	Omejeno	NE	NE	DA	NE	NE
Android	DA	DA	NE	od 3.0	NE	Delno
Windows Phone 7	NE	NE	NE	NE	DA	NE
BlackBerry	DA	DA	NE	NE	NE	NE
Symbian	NE	DA	NE	NE	NE	NE

Tabela 2.1: Podpora posameznih protokolov na predvajalnikih in platformah

na protokole, ki delujejo preko protokola HTTP, je Adobe vključil podporo za predvajanje z več kvalitetami tudi v svoj starejši protokol RTMP.

2.4 Odjemalci

Diplomsko delo se predvsem opira na izdelavo namenskih aplikacij, ki vključujejo predvajanje videa za mobilne naprave. Vsekakor je predvajanje videa na mobilne naprave možno izvesti tudi z uporabo HTML5, ki pa prav tako še vedno ni podprt na zadovoljivem številu naprav, da bi ga ponudniki video vsebin lahko sprejeli za edini način distribucije.

Večina naprav podpira predvajanje s progresivnim pretakanjem. Podpora za protokole s spremenljivo pasovno širino pa je zelo odvisna od predvajalnika in naprave oziroma operacijskega sistema, ki teče na njej. Tabela 2.1 to nazorno prikazuje.

V svetu mobilnega videa prihaja do konsolidacije in združevanja. Adobe ustavlja razvoj predvajalnika Flash za mobilne naprave. HTML5 bo poleg nativnih aplikacij ostal edina možnost za predvajanje videa na mobilnih telefonih in tablicah. V nasprotju z osnovnim HTML5, HLS ne nalaga celotnega videa, kar prinaša prednost. To je velika zmaga za Apple, ki že več let nasprotuje platformi Flash in ne vključuje podpore v svoje naprave. Apple je tako na dobri poti, da njihov protokol za pretakanje videa HLS postane defacto standard za mobilne naprave. Današnja splošna razširjenost protokola HLS je rezultat uspeha operacijskega sistema iOS, ki teče na pametnih telefonih iPhone in tabličnih računalnikih iPad. Apple je protokol postavil kot edini dovoljeni protokol za omenjeni napravi in onemogočil konkurenčne platforme in protokole Flash, Silverlight, RTP, RTSP. Poleg tega tudi enostavno predvajanje s prenosom MP4 datoteke v aplikacijah ni dovoljeno. Zaradi tega mora

vsak večji založnik podpreti protokol HLS, kot ga morajo podpirati vsa glavna orodja za enkodiranje (npr. Sorenson Squeeze) in strežniki (npr. Flash Media Server, Wowza Media Server). Razširjen ekosistem je povzročil, da se podpora protokolu širi tudi na ostale neprenosne naprave kot sta predvajalnika Boxee ali Roku. Podpirata ga tudi konzoli Xbox 360 in Playstation 3. Podpora se razširja tudi na novejše pametne televizorje kot je Samsungov Smart TV. Prav tako je podpora HLS vključena v novejših verzijah operacijskega sistema Android in mnogih drugih.

Za dodatek je lahko vsak posamezen fragment HLS toka podatkov enkriptiran. To je velika prednost v primerjavi s HTML5 videom, ki ga lahko uporabnik predvaja in shrani že s poznavanjem URL video datoteke.

Namera po standardizaciji seveda obstaja in prihaja v obliki MPEG DASH a je še leta stran od splošne implementacije in uporabe na napravah.

Konkurent je protokol Smooth Streaming od podjetja Microsoft, ki pa za delovanje potrebuje vtičnik Silverlight. Tu je tudi HTTP Dynamic Streaming od podjetja Adobe, ki pa potrebuje nameščen vtičnik Flash. HLS je postavljen na vrhu HTML5 tako da omogoča enostavno implementacijo tako na napravah kot v brskalnikih.

Prevlada enega protokola v določenem pogledu niti ni slaba stvar. V nasprotnem primeru bi bili na tem področju priča še večji fragmentaciji. Množica potrebih vtičnikov, različni kodeki in različni protokoli. To je slabo za večje multimedijske korporacije saj zmanjšuje možnost zaslužkov. Še slabše pa je za majhne založnike, ki bi jim pomankanje sredstev onemogočilo ponudbo njihovih vsebin preko mobilnih naprav. Splošen standard pa je še leta daleč od splošne uporabe.

Velika fragmentacija mora biti upoštevana, če želimo video pretakanje ponuditi na različnih platformah. To velja še posebej za predvajanje v živo, kjer kot zasilno možnost ne moremo uporabiti protokola s progresivnim prenosom.

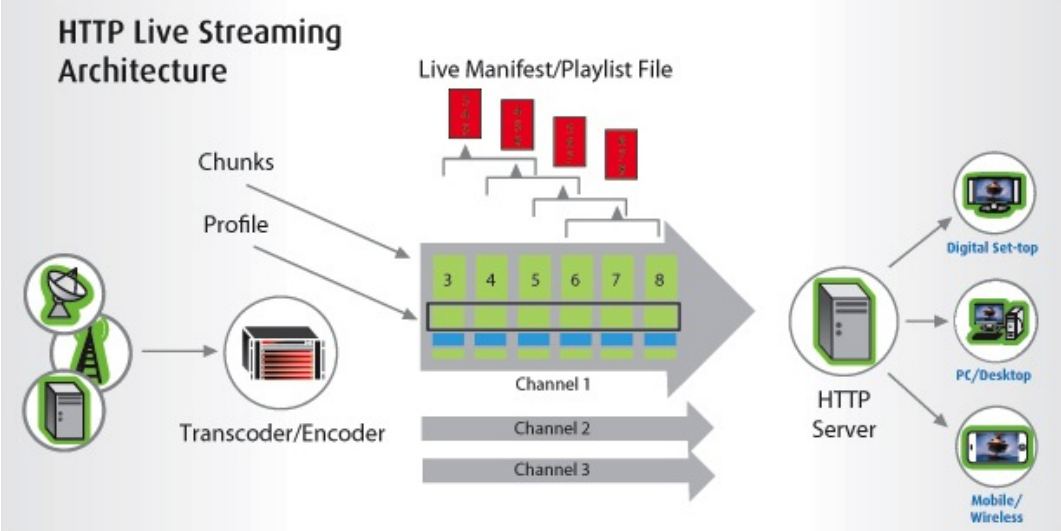
Poglavje 3

Protokol HTTP Live Streaming

3.1 Opis protokola

HTTP Live Streaming (HLS) je komunikacijski protokol za dostavo multimedijskih vsebin, ki temelji na protokolu HTTP. Protokol je razvilo podjetje Apple, predvsem za svoj operacijski sistem iOS kot del programskega ogrodja QuickTime X. V določenih primerih je ta protokol edina možna izbira za dostavo videa v živo na naprave z omenjenim operacijskim sistemom. Protokol HLS je trenutno še v fazi spletnega osnutka, se pa podpora pojavlja na vedno več različnih platformah. Ideja protokola je razbitje toka video podatkov (transport stream) na posamezne segmente in prenos teh segmentov na naprave preko protokola HTTP kot je prikazano na sliki 3.1. Protokol omogoča menjavanje kvalitete predvajanja med predvajanjem brez potrebe po ponovnem zagonu video predvajalnika. Na začetku vzpostavitve vsake seje protokol prenese seznam predvajanja v tekstovni obliki in ga sproti osvežuje. Protokol za delovanje sicer porabi večjo količino prenešenih podatkov, je pa zaradi svoje zasnove na protokolu HTTP odporen na požarne zidove in neodvisen od preusmeritvenih strežnikov. Medijski podatki se lahko prenesejo k odjemalcem kmalu po tem ko so ustvarjeni kar omogoča predvajanje skoraj v živo z manjšim časovnim zamikom.

Protokol ima status internetnega osnutka in je prosto dostopen na straneh IETF[7]. Internet Engineering Task Force je mednarodna organizacija za razvoj in promocijo odprtih internetnih standardov, ki je močno povezana z organizacijami W3C, ISO, IEC. Cilj organizacije je priprava kakovostnih tehničnih dokumentov, ki vplivajo na načrtovanje, uporabo in upravljanje interneta in s tem izboljšati njegovo delovanje. Vodstvo in drugi sodelujoči so



Slika 3.1: Arhitektura protokola HLS

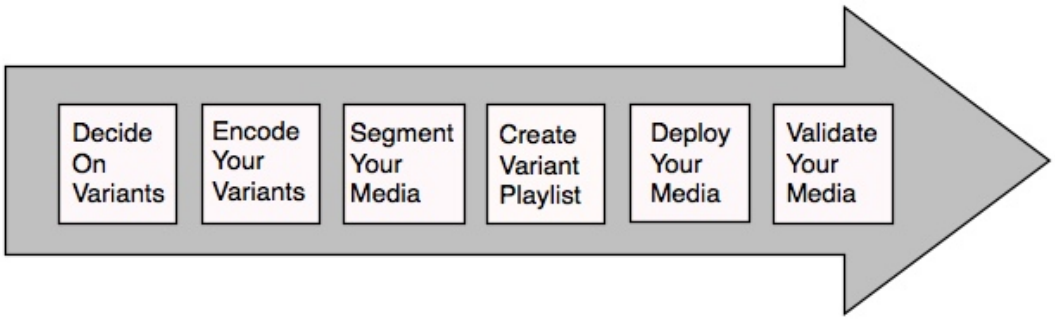
v organizacijo vključeni prostovoljno a je njihovo delo ponavadi financirano oziroma sponzorirano s strani podjetij, kjer so zaposleni. Apple je protokol HLS organizaciji predlagal v obravnavo za internetni standard.

Internetni osnutki (Internet-Drafts) so delovni dokumenti organizacije oziroma njenih delovnih skupin. Skozi razvoj se posamezne specifikacije prosto postavijo v imenik organizacije, ki omogoča prosti dostop in možnost neformalnih pregledov in komentarjev. Tako lajšajo postopke ocenjevanja in revizije. Internetni osnutki nimajo nobenega formalnega statusa in se lahko kadarkoli umaknejo ali spremenijo. Veljavnost posameznega dokumenta je največ 6 mesecev. Dokument se lahko kadarkoli posodobi, nadomesti z drugim dokumentom ali zastari. Odsvetovano je citiranje teh dokumentov v drugih formalnih dokumentih kakorkoli drugače kot "delo v teku". Kot vsi drugi je tudi osnutek protokola za HTTP Live Streaming še v stanju stalnega spreminjanja in dopolnjevanja. Trenutna deveta različica dokumenta opisuje peto verzijo protokola. Objavljena je bila 22.9.2012 in je veljavna do 26.3.2013.

Dokument[3],ki opisuje prokol za prenos neomejenega toka multimedijskih podatkov določa format podatkov in akcije, ki jih opravljajo strežnik in odjemalci oziroma prejemniki podatkov. Opisuje način za enkripcijo medijskih podatkov in možnosti izbire med različnimi verzijami oziroma pasovnimi širinami

Vrsta datoteke	MIME tip
.M3U8	vnd.apple.mpegURL ali application/x-mpegURL
.ts	video/MP2T

Tabela 3.1: MIME tipi, ki jih uporablja protokol HLS



Slika 3.2: Delovni tok priprave video vsebin

istega toka podatkov. Protokol sestavljajo naslednje komponente:

- Datoteke .m3u8 - Tekstovne datoteke, ki vsebujejo metapodatke in poti do video segmentov ali seznamov predvajanja posameznih kvalitetnih nivojev. V primeru da gre za predvajanje neomejene vsebine se ti seznami stalno osvežujejo. Posamezne poti v seznamu kvalitetnih nivojev pa kažejo na posamezne sezname predvajanja, ki so v datotekah iste oblike.

- Datoteke .ts - MPEG 2 Transport Stream. Datoteke z avdio in video vsebino, ki običajno obsegajo za 5 do 10 sekund predvajalnega časa.

Za opisani komponenti je potrebno na strežnikih nastaviti ustrezne tipe MIME, ki so prikazani v tabeli 3.1.

Protokol omogoča samo določeno stopno varnosti. Če uporabnik dobi dostop do naslova URL lahko prenese in deli naprej celotno datoteko. V HLS se lahko vgradi sistem, ki dinamično spreminja varnostne žetone z vsakim novim segmentom, ki je navadno dolg 10 sekund.

3.1.1 Priprava vsebine

Za pripravo vsebine ni mogoče določiti optimalnih parametrov, ki bi vedno veljali, saj so odvisni od mnogih dejavnikov kot je hitrost premikanja objektov glede na ozadje. Poleg tega je odvisno na katere naprave meri ponudnik vsebin. Ker lahko hitro presežemo vsebinske okvire diplomskega dela, se naslednji odstavki večinoma nanašajo le na korake, ki vplivajo na pripravo vsebin za predvajanje s prilagodljivo pasovno širino. Način priprave vsebine tudi ni zajet v dokumentu protokola. Kodiranje se pravzaprav niti ne razlikuje dosti od priprave vsebine v obliki posamezne datoteke, ki ni namenjena pretakanju.

Protokol HLS razen transportnega protokola, ki ga uporablja - MPEG-2 TS, ne določa zvočnega in video kodeka. A če želimo vsebino pripraviti za operacijski sistem iOS se moramo poslužiti standarda H.264/MPEG-4. Kodek H.264 glede na ostale tako ali tako prinaša najboljše rezultate pri enaki pasovni širini, poleg tega pa je pogosto na voljo strojna podpora dekodiranju na napravah uporabnikov.

Prvi od korakov na shemi 3.2 pri pripravi video vsebin za protokol HLS je odločitev o številu kvalitetnih nivojev. Priporočljivo je slediti naslednjim smernicam:

- V primeru, da imamo izvor v ločljivosti HD in želimo podpreti vrsto naprav od telefonov do televizorjev je priporočljivo imeti vsaj en kvalitetni nivo za vsako končno ločljivost videa na podprti napravi. Pogosto tako ponudniki vsebin najprej določijo velikosti okvirjev v katerih se bo video predvajal in glede na to določijo primerne ločljivosti in pasovno širino za posamezen kvalitetni nivo.

- Če je vir v kvaliteti standardne ločljivosti so ponavadi dovoljj trije do štirje kvalitetni nivoji.

- Vsebina, ki je namenjena fiksni velikosti okna naj ima pri vseh kvalitetnih nivojih isto ločljivost in različne pasovne širine ter različno število sličic na sekundo.

- H.264 med kodiranjem vse okvirje razdeli na 16 x 16 blokov. Zaradi tega naj bo resolucija po modulu 16, da se zagotovi najboljšo možna učinkovitost pri stiskanju. Apple v svojih priporočilih tega sicer ne upošteva. Pri večjih resolucijah video vsebin to res niti ni zelo pomembno saj prihranki pri pasovni širini niso opazni.

- Če gre za zabavne vsebine je potrebno več kvalitetnih nivojev. Pri učnih vsebinah potrebe po množici kvalitetnih nivojev ni.

Povezava	Dimenzije	Skupna pasovna širina	Video pasovna širina	Keyframes
Cellular	480 x 320	64 kpbs	audio only	none
Cellular	480 x 224	150 kpbs	110 kbps	30
Cellular	480 x 224	240 kpbs	200 kbps	45
Cellular	480 x 224	440 kpbs	400 kbps	90
WiFi	640 x 360	640 kpbs	600 kbps	90
WiFi	640 x 360	1240 kpbs	1200 kbps	90
WiFi	960 x 540	1840 kpbs	1800 kbps	90
WiFi	1280 x 720	2540 kpbs	2500 kbps	90
WiFi	1280 x 720	4540 kpbs	4500 kbps	90

Tabela 3.2: Priporočila za kodiranje različnih kvalitet podjetja Apple

- Vsak dodaten nivo seveda pomeni dodatne stroške zaradi potrebe po enkoderjih in dodatnem diskovnem prostoru.
- Priporočljiva količina kvalitetnih nivojev naj bo vedno takšna, ki zagotavlja kar največje pokritje naprav uporabnikov za katere je namen ponujati vsebine.
- Obstaja tudi možnost kvalitetnega nivoja ki vključuje izključno zvočno sled.

Pri kodiranju moramo določiti tudi profil, ki določa algoritme stiskanja in vpliva na zahtevnost dekodiranja. Obstajajo osnovni *Baseline* in visoki *High* profili. Za naprave iOS se uporabi *Baseline* profil, ki sicer daje najnižjo možno kvaliteto, a omogoča enostavno dekodiranje in predvajanje. *High profile* po drugi strani daje najvišjo možno kvaliteto slike a za predvajanje zahteva več procesorske moči. Proizvajalec lahko določi osnovni profil za napravo, ki je omejena s procesorsko močjo. Tako ponudi podporo zahtevnemu kodeku H.264, ponudnik vsebin pa bo lahko vedel s kakšnim kodiranjem lahko zagotovi tekoče predvajanje. Profili so tako priročen način da skupni jezik najdejo proizvajalci naprav na eni strani in založniki na drugi.

Pri izbiri pasovne širine se priporoča, da so med kvalitetnimi nivoji opazne razlike. V nasprotnem primeru algoritmi, ki prilagajajo pasovno širino nimajo dovolj manevrskega prostora in izgubijo na pomenu. Seveda so pri nižjih resolucijah videa te razlike manjše - pri Applu je razlike slabih 90 kbit/s med najnižjima nivojema. Pri višjih kvalitetah pa je ta razlika večja - 2 mbit/s med najvišjima priporočenima nivojema.

	SD (Nizka kvaliteta)	SD (Visoka kvaliteta)	HD
Video kodek	H.264 Baseline Profile	H.264 Baseline Profile	H.264 Baseline Profile
Video resolucija	176 x 144 px	480 x 360 px	1280 x 720 px
Število sličic na sekundo	12 fps	30 fps	30 fps
Video pasovna širina	56 Kbps	500 Kbps	2 Mbps
Audio kodek	AAC-LC	AAC-LC A	AC-LC
Audio kanalov	1 (mono)	2 (stereo)	2 (stereo)
Audio pasovna širina	24 Kbps	128 Kbps	192 Kbps

Tabela 3.3: Priporočila za kodiranje različnih kvalitet podjetja Google

Pri kodiranju poznamo dve možnosti nastavitve pasovne širine: stalno in spremenljivo. Spremenljiva pasovna širina sicer ponuja boljšo kvaliteto pri zahtevnih kadrih hkrati pa ne zapravlja nepotrebne pasovne širine pri statičnih kadrih. Tehnologija prilagajanja pasovne širine deluje najboljše takrat ko je menjav med posameznimi nivoji čim manj. V primeru, da so segmenti videa enakih velikosti bo lahko algoritem našel pravi kvalitetni nivo za gladko predvajanje. Tako se v tem primeru za boljšo izkaže odločitev za stalno pasovno širino posamezne kvalitete. Algoritmi bodo pravilneje zaznali zunanjo omejitev pri prenosu segmentov ali možnost preskoka na višjo kvaliteto ob veliki hitrosti prenosa. Pri spremenjivi pasovni širini bi po drugi strani večkrat prihajalo do menjave kvalitetnega nivoja po nepotrebnem samo zato, ker je med predvajanjem prišlo do sekvence, ki je bila težka za kodiranje in je bila potrebna višja pasovna širina. V teh primerih bi bile menjave bolj posledica tehnike kodiranja posameznega videa kot pa sprememb v hitrosti omrežne povezave. Zaradi tega večina strokovnjakov priporoča stalno pasovno širino, ki žal prinaša slabšo sliko pri zahtevnih kadrih ali spremenljivo, ki pa je omejena z najvišjo in najnižjo pasovno širino in med njima ni prevelike razlike.

Ker vse tehnologije pretakanja s prilagodljivo pasovno širino uporabljajo neko vrsto segmentiranja video toka na posamezne dele, je pomembno, da se vsak segment začne s ključnim okvirjem (*keyframe*), kot je navada tudi pri drugih načinih predvajanja videa. *Keyframe* je edini okvir, ki vključuje vse potrebne informacije za dekodiranje in predvajanje. Da se zagotovi keyframe na začetku vsakega segmenta je potreben stalen interval med segmenti v množici kvalitetnih nivojev. V orodjih za kodiranje to onemogoči postavitev keyframe okvirjev glede na menjave kadrov v vsebini. H.264 kodiranje za iOS naj bo

tako narejeno v *single-pass Baseline Profile*, onemogočena naj bo reorganizacija video okvirjev. Ključni okvirji pa se po priporočilih naj pojavijo na vsakih 5 sekund, v idealnem primeru naj bo čas sodi delitelj izbrane dolžine segmentov.

Nespremenljivost parametrov za kodiranje zvoka priporočata tako Apple kot Google. Enaka pasovna širina zvočne sledi pri vseh kvalitetnih nivojih zagotavlja, da se zvok gladko predvaja pri preskokih med posameznimi kvalitetami. Apple za zvočno sled priporoča 40 kbps pri vseh kvalitetnih nivojih. Očitna slabost tega je, da se skupaj s kvaliteto slike ne povečuje tudi kvaliteta zvoka, kar je še posebej pomembno pri prenosu koncertov in podobnih dogodkov, kjer je kakovost zvoka pomemben faktor. Poleg tega je bistvena razlika, če se medijska vsebina predvaja na mobilni napravi ali na hišnem kinu. Če zvok ni zelo pomemben je bolj varno uporabljati enotno nastavitev za zvočno kodiranje pri različnih kvalitetnih nivojih. Če pa je kvaliteta zvoka zelo pomembna, je ena od možnosti uporaba *mono* in *stereo* tehnike. Za nizke kvalitetne nivoje se naprimer uporabi 64Kbps/44 kHz/16-bit mono, za visoke pa naprimer 128Kbps/44 kHz/16-bit stereo. Tako se zmanjša možnost, da bo pri preskoku med kvalitetami prihajalo do napak v zvoku.

Najti je možno različna priporočila za konfiguracijo kvalitetnih nivojev a podjetje Apple ponuja najbolj natančna. Priporočene kvalitetne nivoje, ki seveda najbolj ustrezajo napravam z njihovim operacijskim sistemom, lahko najdemo v tabeli 3.2. Hkrati se zahteva[2] tudi dodatne omejitve za aplikacije na njihovi platformi. Video mora nujno vsebovati kvalitetni nivo s pasovno širino, ki ni večja od 64 kbps, če bo aplikacija omogočala predvajanje tudi preko mobilnih omrežij.

Google po drugi strani navaja bolj splošne napotke in ne zahteva posebnih minimalnih kvalitetnih nivojev, kot se lahko vidi v tabeli 3.3.

3.1.2 Segmentiranje

Po kodiranju vsebin v ustrezne kvalitetne nivoje je na vrsti segmentiranje, priprava seznamov predvajanja in drugih metapodatkov ter validacija pripravljenih medijskih vsebin. Eden izmed lažjih načinov za protokol HLS je uporaba orodij podjetja Apple, bolj zahtevni uporabniki pa lahko orodja najdejo pri proizvajalcih strežnikov in sistemov za pretok videa. Prosto dostopnih je tudi nekaj odprtokodnih knjižnjic, ki izhajajo iz FFMpeg. Za segmentiranje toka podatkov je možen tudi lasten sistem, ki indeksira tok podatkov, shrani posamezne kose v obliko .ts datotek ter pripravi ustrezne sezname predvajanja

.m3u8.

Proces segmentiranja pripravi vsebino v formatu H.264/AAC kot sekvenco manjših kosov, ki so v formatu MPEG-2 TS. Vse kvalitete predvajanja morajo biti segmentirane v kose ob točno enakih časih, da se omogoči neopazen prehod med kvalitetami med predvajanjem. Zraven se ustvari indeksna datoteka M3U8, ki hrani reference na vse segmente. V primeru živega toka podatkov se sproti posodablja, da vedno odraža realno stanje.

S krajšo dolžino segmenta dosežemo manjši zaostanek, če predvajamo živo sliko. Z daljšo storimo ravno obratno. V vsakem primeru morajo biti po specifikaciji v seznamu predvajanja vedno na voljo vsaj trije segmenti. Praksa in tudi različna priporočila kažejo, da je optimalna dolžina segmenta okrog 10 sekund. Če moramo imeti v vsakem trenuku vsaj 3 segmente na voljo za predvajanje pomeni, da je običajen minimalni zaostanek za živo sliko 30 sekund.

Pri številu segmentov v premikajočem oknu moramo prav tako najti zlato sredino. Z dolgim seznamom predvajanja z množico segmentov bi sicer zmanjšali potrebo po stalnem osveževanju a bi po drugi strani povzročili dolgo zakasnitev za živo sliko. Naslednja prednost bi bila večja odpornost na prekinitve, ki pa ni bistvena, saj brez povezave tudi video segmentov ni mogoče prenesti in je edina možnost prekinitev predvajanja. Stalno osveževanje indeksnih datotek v primerjavi z video podatki še vedno predstavlja zanemarljiv delež prenosa. V praksi se tako največkrat uporabijo samo trije segmenti v premikajočem oknu.

3.1.3 Seznami predvajanja

Multimedijska vsebina je določena z URI naslovom do datoteke seznama predvajanja, ki je urejen seznam URI povezav skupaj z informacijskimi oznakami oziroma meta podatki. URI naslovi in priložene oznake sestavljajo serijo multimedijskih segmentov.

Za predvajanje multimedije odjemalec pridobi datoteko s seznamom predvajanja in nato prenese in predvaja vsakega od medijskih segmentov. V primeru živega toka podatkov lahko odjemalec osvežuje datoteko s seznamom predvajanja in tako pridobi dodatne segmente, ki se sproti pojavljajo v seznamu predvajanja in so na voljo za predvajanje.

Indeksne datoteke so kodirane v znakovnem formatu UTF-8. m3U8 je razširjen format M3U seznama predvajanja. Predlog protokola HLS razširja obstoječi M3U z določitvijo dodatnih oznak. Seznam je tekstovna datoteka, ki

sestoji iz posameznih vrstic, ki so lahko naslov URI, prazne ali pa se začnejo z znakom #. Prazne vrstice se ignorirajo, nevidni znaki kot je presledek pa se smejo pojavljati samo tam, kjer je to izrecno dovoljeno. Vrstica z URI skupaj s pripadajočimi oznakami določa posamezen multimedijski segment.

Vrstice, ki se začnejo z znakom '#' so komentarji ali oznake. Vsaka oznaka se začne z #EXT. Vse ostale oznake so komentarji in jih mora odjemalec ignorirati. URI lahko določa pot do datotek absolutno ali relativno. Če je URI relativen se razreši glede na URI datoteke v kateri je vsebovan. Dolžina seznama predvajanja je enaka vsoti dolžin vseh multimedijskih segmentov v seznamu.

Indeksne datoteke so lahko dolge in se ovežujejo precej pogosto, po drugi strani pa so to tekstovne datoteke, katerih stiskanje je zelo učinkovito. Protokol tako predlaga tudi vključitev podpore za stiskanje vsebine gzip. Gzip stiskanje uporablja algoritem DEFLATE, ki je kombinacija LZ77 in Huffmanovega kodiranja. Za vključitev stiskanja gzip je potrebno na strežniku to omogočiti. Strežnik bo tako v glavo odgovora HTTP dodal Content-Encoding: gzip, vsa vsebina pa bo sproti stisnjena. Če implementiramo svoj predvajalnik pa vsem zahtevkom HTTP v glavo dodamo če vrstico Accept-Encoding: gzip. Pri dolgih .m3u8 indeksnih datotekah je lahko stiskanje tudi 10 x[4], kar prinese prihranek v količini prenešenih podatkov, ki je pomemben predvsem pri uporabi mobilnih omrežij z nizko pasovno širino ali v primerih ko uporabnik plačuje omrežno povezavo glede na količino prenešenih podatkov. Seveda stiskanje seznamov predvajanja po drugi strani s seboj prinaša tudi večjo potrebo in zahtevnost po sistemskih virih, tako na strežniški kot na odjemalčevi strani.

Vrstni red posameznih nivojev predvajanja v datoteki je glede na zastavljen protokol nepomemben. Ponavadi so postavljeni od najnižje do najvišje pasovne širine. Večina predvajalnikov začne s predvajanjem prve kvalitete s seznama. Mnogokrat se ponudniki vsebin tako odločijo, da na prvo mesto postavijo kvaliteto, ki se najbolj pogosteje predvaja.

Za strežbo teh datotek je primeren vsak standardni HTTP strežnik. Ker ni potrebe po namenskih srežnikih za pretočni video lahko uporaba protokola HLS za ponudnika pomeni finančni prihranek.

3.1.4 Predvajanje na napravah

Podpora različnim napravam se skozi čas nenehno povečuje. Če je bil protokol sprva ustvarjen samo za operacijski sistem iOS, se je sedaj razširil na najrazličnejše vrste naprav. Poleg mobilnih naprav kot so pametni telefoni in tablice se vedno bolj pojavljajo implementacije za različne platforme. Podporo HLS izpostavljajo tudi vsi glavni proizvajalci pametnih televizorjev, ki že vključujejo tudi možnost povezave na internet. Programski vmesnik za uporabo protokola ponujajo zunanjim razvijalcem za razvoj aplikacij za njihove televizorje. Prav tako je podpora protokolu na voljo v različnih multimedijskih predvajalnikih in IPTV vmesnikih. Če podpora protokola na določeni napravi ni na voljo že preko osnovnega sistema ali programskega vmesnika pa je tukaj precej predvajalnikov ali knjižnjic zunanjih proizvajalcev, ki to podporo imajo.

Razvijalci lahko uporabljajo protokol na dva različna načina. Na večini platform je na voljo programski vmesnik, ponavadi imenovan *MediaPlayer*, preko katerega se lahko uporablja množica različnih protokolov in kodekov. Če programskemu vmesniku določimo pot do .m3u8 datoteke jo ta samodejno prepozna kot predvajanje s protokolom HLS. Dodatni klici, ki jih takšni vmesniki premorejo, so poleg standardnih video ukazov kot sta *premor* in *stop* še ukazi, s katerimi lahko razijalec določi najvišjo, najnižjo ali začetno pasovno širino predvajanja in tako izboljša uporabniško izkušnjo. Pri mobilnih platformah lahko razvijalec ugotovi, če gre za zmogljivo napravo povezano na hitro Wi-Fi omrežje in tako nastavi višji začetni kvalitetni nivo. V drugem primeru lahko naprimer uporabniku omogoči nastavitev zgornje omejitve pasovne širine in tako prepreči prevelik prenos podatkov preko mobilnih omrežij.

V primeru, da je podpora protokolu na voljo tudi v spletnem brskalniku, se video vsebina lahko enostavno vključi v spletne aplikacije s HTML 5 video oznako[2]:

```
<html>
<head>
    <title>Primer HTTP Live Streaming</title>
</head>
<body>
    <video
     src="http://devimages.apple.com/iphone/samples/bipbop/bipbopall.m3u8"
     height="300"
     width="400">
    </video>
```

```
</body>
</html>
```

3.2 Glavne značilnosti protokola

3.2.1 Prednosti

- Prilagajanje pasovne širine - protokol se samodejno prilagodi glede na zastoje pri prenosu in razpoložljivo pasovno širino.
- Odpornost na prehodne omejitve pri omrežni povezavi.
- Ni potrebe po dodatnih nastavitvah na strežnikih, usmerjevalnikih ali na požarnih zidovih. Vse poteka preko protokola HTTP 1.1, uporabi se lahko navaden HTTP strežnik.
- Enostavna integracija v omrežja za dostavo vsebin (CDN).
- Podpora predvajanju v živo.
- Podpora protokolu v prihajajočen HTML5 z uporabo video oznake (video tag) na napravah kot je Google TV.
- Možnost uporabe AES enkripcije.

3.2.2 Slabosti

- Protokol še vedno ni podprt pri nekaterih proizvajalcih mobilnih naprav in v najbolj razširjenih operacijskih sistemih za osebne računalnike kot je Microsoft Windows.
- Nepopolna podpora pri sistemih, ki naj bi ta protokol podpirali. Tukaj je sicer krivda na strani proizvajalcev naprav, ki ne ponudijo posodobitev na novejše različice Android OS a vseeno je to omejitev, ki pade na pleča ponudnikov vsebin.
- Precejšnja količina režijskih podatkov v primerjavi z drugimi protokoli kot je RTSP. Stalno osveževanje prinaša precejšno količino prenosa metapodatov.

- Starejša tehnologija ovira pot k navišji možni učinkovitosti in ekonomičnosti pri količini prenešenih podatkov. Protokol za pakiranje uporablja MPEG-2 TS transportni protokol, ki je v uporabi že več kot 10 let in za dostavo paketov uporablja protokol ATM. Ta protokol je bil narejen za čas pred omrežji IP in s sabo nosi v glavi 4 bajte pri vsakem 188 bajtov velikem paketu. Čeprav se to ne zdi veliko, se pri eni uri gledanja pri 1,5 mbit pasovne širine prenese za 115000 kilobitov nepotrebnih podatkov, ki poleg tega zahtevajo nekaj procesorske moči za odpakiranje. Druga nepotrebna funkcionalnost protokola MPEG-2 TS je tudi multipleksiranje različnih zvočnih tokov v en podatkovni tok. Ker uporabnik v določenem trenutku posluša največ en zvočni kanal, bi lahko to bilo urejeno programsko.

3.3 Konkurenčni protokoli

Poleg opisanega protokala HLS sta trenutno na trgu še dva množično uporabljena produkta, ki se uporabljata v industriji. Prvi je Microsoft Smooth Streaming, ki je bil prvi na trgu uporabljen med ponudniki vsebin in se lahko uporablja za večino platform. Drugi je Adobe HTTP Dynamic Streaming (HDS), ki je del multimedijske platforme Adobe Flash in je še posebej priljubjen med ponudniki video vsebin za splet.

Izbira protokola za dostavo vsebin največkrat zavisi od izbire platform, za katere želi ponudnik vsebin imeti podporo za svoje produkte. V primeru, da ponudnik želi podpreti naprave z operacijskim sistemom iOS je omejen s politiko podjetja Apple za njihovo trgovino aplikacij App Store. Če traja predvajanje posnetkov več kot 10 minut ali je je količina prenešenih podatkov v periodi 5 minut več kot 5MB preko mobilnih omrežij, kakršenkoli drug protokol razen HLS ni dovoljen. Microsoft pa po drugi strani podpira samo Microsoft Smooth Streaming za naprave z njihovim operacijskih sistemom Windows Phone 7. Pri večini drugih naprav ni omejitev pri pravilih pri izdelavi aplikacij za njih a kljub temu določeni protokoli prav tako niso podprti.

V tabeli 3.4 je primerjava omenjenih protokolov.

3.3.1 Microsoft Smooth Streaming

Smooth Streaming je razširitev platforme za dostavo medijskih vsebim IIS Media Services, ki omogoča pretakanje video vsebin preko protokola HTTP s prilagajanjem pasovne širine. Microsoft je to tehnologijo najavil leta 2008 kot del tehnologije Silverlight. Že istega leta so demonstrirali prototip rešitve ko so

Comparison of Approaches for Adaptive HTTP Streaming Technology

	3GPP/MPEG DASH	Apple HTTP Live Streaming	Microsoft Smooth Streaming	Adobe HTTP Dynamic Flash Streaming	3GPP RTSP Streaming
Type	Open, standards-based	Single-vendor controlled	Single-vendor controlled	Single-vendor controlled	Open, standards-based
Source Video Codecs	H.264 + others (agnostic)	H.264	H.264, VC-1	H.264, VP6	H.263, H.264, MPEG-4
Source Audio Codecs	AAC + others (agnostic)	AAC, MP3	AAC, WMA	AAC, MP3	AAC, AMR
Package/Segment format	MP4 fragments + MPEG-2 TS*	MPEG-2 TS	MP4 fragments	MP4 fragments	None (RTP packets)
File storage on server	Contiguous or individual files per segment	Individual file per segment (pre iOS 5.0)	Contiguous	Contiguous	Contiguous
Audio/video/text packaging	Multiplexed or separate segments for audio, video	Multiplexed in 1 segment (pre iOS 5.0)	Multiplexed in 1 segment	Multiplexed in 1 segment	Separate streams for a/v
Segmentation and Delivery	Multiple vendors. Standard HTTP or Streaming servers (+ Helix)	Multiple vendors. Standard HTTP or Streaming server (+Helix)	MS IIS (+few other vendors incl Helix)	Adobe Interactive Server (or Adobe tools + Apache module for on-demand)	Multiple vendors for packaging and streaming servers
Playback	3GPP-Rel 9 or MPEG clients (+ Helix)	Apple iOS , QT X (+ Helix)	Silverlight (+ Helix)	Flash, Air	3GPP Handsets (+ Helix)
Protection	Flexible (e.g., OMA or UV)	AES-128 encryption	PlayReady	Flash Access	OMA
Typical Segment Duration	Flexible	10 sec	2-4 sec	2-4 sec	None (RTP packets)
Adaptation Control	Client	Client	Client	Client	Server

Slika 3.3: Primerjava različnih tehnologij pretakanja videa

v živo v obliki videa na zahtevo ponudili možnost predvajanja olimpijskih iger v Pekingu. Specifikacija bazira na datotečnem formatu ISO Base Media File Format, ki definira splošno strukturo za časovno odvisne multimedijske datoteke kot sta video in audio. Microsoft določa obliko s specifikacijo formata PIFF. Podjetje je aktivno vključeno v organizacije 3GPP, MPEG in DECE in si prizadeva za standardizacijo teh tehnologij. Za platformi Silverlight in Windows Phone 7 Microsoft zagotavlja razvijalno okolje za vse, ki si želijo implementirati to tehnologijo v svoje aplikacije. Microsoft zagotavlja dostavo tako videa na zahtevo kot živih posnetkov pri visoki kvaliteti 1080p za odjemalce s Silverlight podporo. Tehnologija se lahko uporabi tudi za pretakanje videa pri spletnih rešitvah z uporabo vtičnika. Smooth Streaming trenutno podpira kodeka VC-1 in H.264. V privzeti konfiguraciji IIS Smooth Streaming kodira video tok v sedem različnih kvalitet v rangu od 300 kbps do 2,4 Mbps pasovne širine.

Za druge odjemalce kot so naprave z Apple iOS, Android in Linux operacijskimi sistemi pa Microsoft ponuja paket za pretvorbo v oblike, ki jih podpirajo odjemalci omenjenih operacijskih sistemov. IIS Media Services 4.0, ki je bil izdan v novembru 2010, je predstavil funkcionalnost, ki lahko sproti prepakira video Smooth Streaming kodiran z H.264/AAC v format HLS in tako omogoči dostavo na naprave iOS brez potrebe po posebnem rekodiranju. To velja tako za video na zahtevo kot za prenose v živo.

Smooth Streaming vključuje vse tipične karakteristike pretakanja s prilagodljivo pasovno širino. Video je segmentiran v manjše kose, ki so dostavljeni odjemalcem preko protokola HTTP. Ponavadi se medijski tok kodira v več različnih kvalitet, da lahko algoritem na odjemalcu izbere najboljšo video kvaliteto glede na omejitve pri pasovni širini in tako uporabnikom ponudi optimalno kvaliteto za ogled. Glede na vse pozitivne lastnosti takšnega načina, ki so naštete v drugem poglavju si Microsoft močno prizadeva za uveljavitev svoje družine tehnologij Silverlight, Smooth Streaming in Mediaroom.

V nekaterih delih se je Microsoft odločil za drugačne pristope pri implementaciji, ki določajo nekaj ključnih razlik med njihovim protokolom Smooth Streaming in Applovim HLS. HLS redno osvežuje premikajoče okno metapodatkovnih datotek, ki odjemalcu povedo, kateri segmenti so na voljo za prenos. Smooth Streaming pa uporablja časovne oznake pri zahtevah za segmente tako da ponavljajoče prenašanje indeksnih datotek ni potrebno. To pripelje do ključne razlike: Ker HLS zahteva osvežitev seznama predvajanja ob vsakem novem segmentu, ki je na voljo, je splošna praksa, da se uporablja daljše segmente, da se zmanjša število zahtev po osvežitvi seznama predvajanja. Ta

čas je priporočeno 10 sekund medtem ko je pri protokolu Smooth Streaming 2 sekundi.

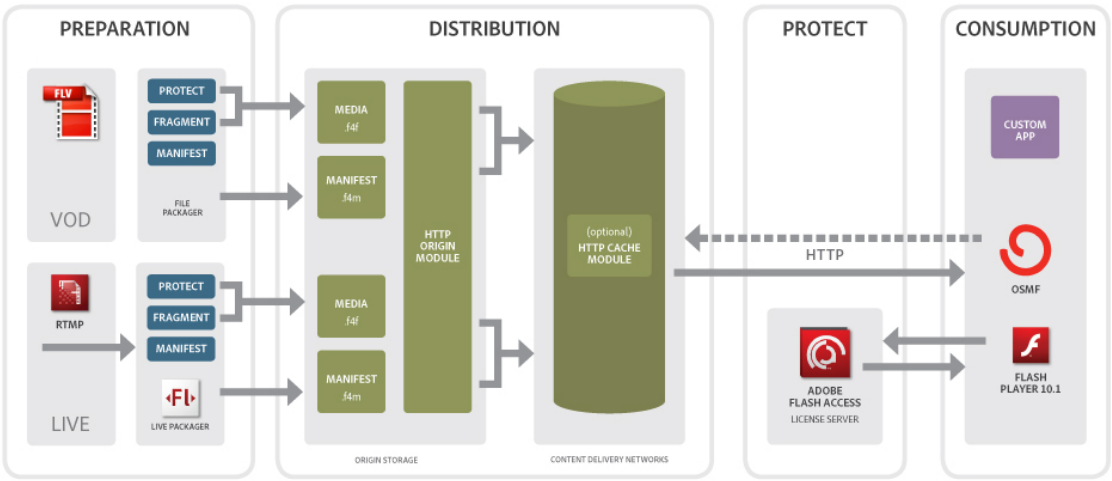
Oba formata sicer uporabljata H.264 video kodiranje in AAC audio kodiranje a je pri HLS ta tok ovit v MPEG-2 TS Transport Stream datoteke, pri protokolu Smooth Streaming pa se uporabljajo fragmenti ISO MPEG-4 datotek. Fragmenti MP4 datotek so oblika pri kateri vsi podatki iz osnovne datoteke niso vključeni v datoteki za dostavo odjemalcu. Obe obliki imata nekaj prednosti in slabosti. MPEG-2 TS datoteke pri HLS vsebujejo množico orodij za analizo in imajo predefinirane mehanizme za signalne podatke. Fragmentirane MP4 datoteke so po drugi strani bolj fleksibilne in se lahko hitro prilagodijo za različne vrste podatkov kot so informacije za dekriptiranje videa. MPEG-2 TS datoteke takšnih rež ne vsebujejo.

Razlika je prav tako v potrebni podpori infrastrukture za ta dva formata. Microsoft je vključil podporo za Smooth Streaming v njihov strežnik IIS. Tak strežnik shrani datoteko Smooth Streaming v agregirani obliki kar pomeni manjše število datotek, v katerih so zbrani vsi segmenti. Kljub temu, da lahko tudi HLS podpira takšno hranjenje datotek ta funkcionalnost ni podprta s trani Appla. Microsoft ravno to možnost hranjenja datotek navaja kot močan argument v primerjavi HLS, saj je lažje upravljanje z eno neprekinjeno datoteko kot s tisoči datotek za posamezen video pri protokolu HLS. To sicer drži gledano s strani strežnika, ki streže vsebino, s strani spletnega strežnika ali večjega omrežja za dostavo vsebin CDN pa so zahtevki definirani še vedno za posamezne segmente in se prav tako k odjemalcem prenesejo kot ločeni kosi video toka podatkov.

Slabost pristopa z eno samo datoteko za celoten video tok je več potrebnega dela za strežnik ko odjemalec zahteva določen segment. Kot pri HLS mora odjemalec najprej pridobiti meta datoteko, ki je v tem primeru imenovana .ismc. Ta datoteka vključuje informacije o tem, kateri tokovi so na voljo, kakšne so pasovne širine, kodeki itd. in določa s kakšnimi URL zahtevki se pride do njih. Zaradi uporabe neprekinjene datoteke za posamezen kvalitetni tok pa pride do dveh razlik v v arhitekturi strežnik-odjemalec:

- Odjemalec prebere meta datoteko in zahteva posamezen segment s parametrom časovnega žiga in ne unikatnega URL naslova kot pri HLS.

- Strežnik mora iz parametrov zahtevka izračunati in določiti odmik in obseg bajtov v celotni datoteki ter jo kot segment poslati odjemalcu.



Slika 3.4: Shema delovanja protokola Adobe HTTP Dynamic Streaming

3.3.2 Adobe HTTP Dynamic Streaming

Tehnologija podjetja Adobe je na spletu osnovana storitev, ki je na voljo za vse naprave, ki poganjajo spletni brskalnik in imajo nameščen vtičnik Adobe Flash. Adobe je rešitev izdal v sredini leta 2010 in želi kot zadnji izmed velikih treh nadoknaditi zaostanek. Ker je glavna platforma za odjemalce Flash, se HDS dostikrat omenja tudi pod imenom HTTP Flash Streaming. HDS je tipični predstavnik protokola za pretok videa s prilagodljivo pasovno širino. Video vsebine so segmentirane v manjše kose, ki se nato dostavijo preko protokola HTTP. Ker so vsebine kodirane v več kvalitetnih nivojev lahko odjemalec med njimi preklaplja in uporabniku zagotovi optimalno kvaliteto glede na omejitve pri omrežni povezavi ali procesorski moči.

Tudi Adobe za svojo rešitev ponudnikom vsebin zagotavlja mehko in za uporabnike nezaznavno zamenjavo med posameznimi kvalitetami video toka. Njihov algoritem za prilagajanje pasovne širine se opira na omejitve pri mrežni povezavi in sposobnosti strojne opreme odjemalca. Sam delovni tok vključuje orodja za pripravo vsebine, fragmentiranje datotek MP4, ki se lahko predpomnejo, programsko ogrodje za predvajanje na odjemalcih Open Source Media Framework in možnost zaščite multimedijskih vsebin z uporabo programske opreme Adobe Access, ki je podprt od verzije Flash Player 10.1 naprej in omogoča rešitev za stabilno zaščito vsebin in njihovo prodajo. Celotna shema delovanja je prikazana na sliki 3.4.

Predvajalnik Flash in Flash medijski strežnik podpirata pretakanje s prilagodljivo pasovno širino tako preko tradicionalnega protokola RTMP kot tudi protokola HTTP. S slednjim Adobe ponuja podobno rešitev kot njegova konkurenta Apple in Microsoft. HDS podpira video kodeka H.264 in VP6, ki sta že vključena v vtičnik Flash. To predstavlja veliko prednost v primerjavi s konkurenti, saj je razen izjem kot so naprave Apple iOS izjemno razširjen med odjemalci. Kot primer uporabe lahko navedemo spletni video portal YouTube ali predvajalnik iPlayer britanske televizijke hiše BBC. Apple za svoje naprave z operacijksim sistemom iOS ni vključil podpore za Adobe Flash in je s tem onemogočil tudi uporabo HDS.

HTTP Dynamic Streaming je zgrajen in postavljen s pomočjo klasičnega strežnika HTTP kot je Apache. Za izdelavo aplikacij in predvajalnikov pa je na voljo prosto dostopen API (Flash Player) in odprtokodno programsko ogrodje Open source media framework (OSMF) - enako kot pri uporabi progresivnega pretakanja. Vsak obstoječi RTMP podatkovni tok se lahko enostavno pretvori v datotečni format, ki se dostavi preko HTTP. Adobe za HDS uporablja enako tehnologijo za HTTP strežnike kot za pretakanje preko progresivnega protokola a omogoča nekatere dodatne funkcionalnosti:

- Podpora predvajanju s spremenljivo pasovno širino. HTTP progresivno prenašanje po drugi strani porabi večjo pasovno širino, ker celoten video prenese kakor hitro mogoče tudi če si ga gledalec ne bo ogledal do konca.
- Možnost iskanja po časovnem traku daljših video predvajanj, saj mu ga ni potrebno prenesti vnaprej.
- S kontrolo pasovne širine se lahko prepreči, da bi drugim aplikacijam odvzel prevelik del pasovne širine povezave.
- Predvajanje v živo z DVR zaščito vsebin v realnem času z uporabo Adobe Access.

Za dostavo pretakajočih vsebin preko HTTP se celoten tok pošlje skozi proces fragmentacije, kjer se lahko s pomočjo dodatnega modula vsebino tudi zaščiti pred nepooblaščenim kopiranjem in predvajanjem. Protokol podpira tako predvajanje v živo kot video na zahtevo. Za fragmentacijo se uporablja standardni format fragmenta MP4 (F4F). Podprti video in avdio kodeki so VP6/MP3 in H.264/AAC. Kot pri protokolih HLS in SS dominira kodek H.264/AAC.

Podobno kot pri drugih protokolih na začetku predvajanja odjemalec prenese manifestno datoteko, ki je pri HDS datoteka F4M in nosi vse informacije,

ki so potrebne za predvajanje vsebine. Vključuje informacije o avdio in video kodekih, formatu fragmentov, medijskih tokovih, ki so na voljo, kvalitetnih nivojih, ki so na voljo, pot do licenčne datoteke za Flash Access in ostale metapodatke. Datoteke z video vsebino se nahajajo na HTTP strežniku, ki je odgovoren za pošiljanje pravih kosov video vsebine glede na zahteve odjemalcev. S pomočjo Adobe Open Source Media Framework (OSMF) lahko tako že splošno razširjen osnovni HTTP strežnik kot je Apache zadostuje za prenos video vsebin uporabnikom.

V primerjavi s protokolom HLS ima Adobe HDS nekaj ključnih razlik:

- HLS stalno osvežuje premikajoče okno indeksne datoteke, ki predvajalniku pove, kateri kosi so na voljo za prenos. Adobe HDS uporablja številko sekvence v zahtevi strežniku tako da predvajalniku ni potrebno ves čas prenašati indeksne datoteke.

- Poleg indeksne datoteke obstaja še začetna datoteka pri predvajanjih v živo. Ta podaja posodobljene številke sekvenc. Pri HLS se za to uporablja stalno osveževanje indeksne datoteke.

- Ker HLS zahteva osvežitev indeksne datoteke ob vsakem novem segmentu, je smotrno uporabljati daljše segmente da se zmanjša število zahtevkov na strežnik. Zadnje različice odjemalcev za HLS sicer osvežujejo indeksno datoteko šele ko jim zmanjka URL naslovov do segmentov. V primeru da se je premikajoče okno v velikosti treh segmentov po 10 sekund lahko predvajalnik osvežuje indeksno daotteko na nekaj manj kot 30 sekund. A kljub temu je pri HDS dolžina segmenta mnogo krajša, ponavadi od 2 do 5 sekund. Pri HLS je čas segmenta ponavadi 10 sekund.

- Format, ki povezuje posamezne segmente je pri protokolih različen. Oba protokola sicer podpirata H.264 video kodiranje in AAC avdio enkodiranje a HLS za transport uporablja datoteke MPEG-2 Transport Stream, Adobe HDS pa tako kot Microsoft SS uporablja fragmentirane ISO MPEG-4 datoteke.

- Ker je pri HDS video tok na strežniku še vedno v eni datoteki mora strežnik iz zahtevka izluščiti segment in izračunati odmik kje točno se zahtevani segment nahaja in ga posredovati naprej odjemalcu.

HDS ima tako več skupnega z Microsoftovim protokolom Smooth Streaming kot Applovim HLS. Primarno zato, ker uporablja eno datoteko iz katere se generirajo segmenti v obliki datoteke MPEG, ne pa individualne segmente kot jih uporablja protokol HLS.

3.3.3 Druge sorodne tehnologije

Duge možnosti za dostavo videa na pametne telefone in tablične računalnike, ki vsebujejo možnost prehodov med različnimi kvalitetami predvajanja in s tem pasovnimi širinami so še:

- Octoshape Multi-BitRate podpira pretakanje z večimi pasovnimi širinami vsebine. Za vhod uporablja standardne pretočne formate kot je Flash, Windows ali HLS. Octoshape ponuja edinstveno tehnologijo optimizacije pretoka in elastično kodiranje sheme, da maksimizira konsistentnost pasovne širine video toka preko interneta. Kot drugi tudi Octoshape podpira preskoke med različnimi kvalitetami a različno od na TCP protokolu osnovanih tehnologij kot sta HTTP ali RTMP, ki imata spreminjajoč se pretok zaradi omrežnih latenc in zakasnitev. Algoritem izbere primerno pasovno širino na začetku predvajanja in kasneje med predvajanjem redko preklopi med kvalitetami ter tako uporabnikom prinaša nespremenljivo uporabniško izkušnjo. Octoshape je prav tako prva tehnologija, ki uporablja samodejno večpasovno pretakanje pri naslavljanju mnogim prejemnikom preko interneta hkrati (multicast).

- Quavlive Adaptive Live Streaming oziroma QuavStreams, ki ga je razvilo podjetje Quavlive s.r.l. omogoča množičo spletno distribucijo video vsebin. Poleg kodeka H.264 podpira novi kodek WebM. Tudi ta rešitev vključuje možnost preklapljanja med video kvalitetami in podpira poljubno število različnih pasovnih širin. Privzete pasovne širine se raztezajo od 300 kbps do 2.4 Mbps in se lahko nastavljajo v času enkodiranja. Območje resolucij videa pa se razteza od 320x240 do 1280x720. Za razliko od drugih Quavlive Adaptive Streaming uporablja drug pristop za prilagajanje kvalitete med predvajanjem in sicer na strani strežnika. To prinaša prednost, saj je ob nadgradnji logike za prilagajanje potrebna samo nadgradnja na strani strežnika. Za izbiro najboljše kvalitete predvajanja uporablja algoritem prilagajanja brez dodatne faze predpomnenja ali nastavitev uporabnika. Po njihovih navedbah je aplikacija QuavStreams prva, ki vsebuje podporo za pretakanje s prilagodljivo pasovno širino za HTML5.

Poglavje 4

Implementacija protokola HLS v programskem jeziku Java

Google je v operacijski sistem Android podporo za HLS prvič vključil šele v verziji 3.0. Poleg tega tudi ni bilo točno določeno v kolikšni meri je protokol dejansko podprt. Glede na testiranja in prve prototipe se je izkazalo, da je implementacija zelo slaba oziroma podpore sploh ni bilo. Na napravah kjer je podpora bila smo se srečevali z različnimi problemi. Ker je bila ob pričetku razvoja aplikacije mobilne televizije podpora novejšega rezličice sistema na napravah uporabnikov praktično nična je bila smiselna tudi zahteva naročnika, da se podpre tudi naprave z verzijo operacijskega sistema 2.2 in 2.3.

4.1 Uporabljena tehnologija in orodja

4.1.1 Programski jezik Java

Z zahtevami po podpori platforme Android in vseh glavnih operacijskih sistemov za osebne računalnike je bil programski jezik Java logična izbira. Ob začetku so si ustvarjalci Jave zadali 5 ciljev, med katerimi so tudi lastnosti, ki so pripomogle pri implementaciji protokola: preprostost, objektna orientiranost, robustnost, neodvisnost od arhitekture in prenosljivost ter večnitnost.

Edina druga možnost na platformi Android bi bila uporaba Native Development Kit, kjer se uporablja programski jezik C. S stališča porabe procesorske moči je delovanje protokola v primerjavi z dekodiranjem videa in izrisovanjem na zaslon zanemarljiv proces. Tako ni bilo nobene potrebe po večji učinkovisti

programa, ki bi ga prinesla uporaba razvojnega orodja NDK.

4.1.2 Android

Android je odprtokodni operacijksi sistem, ki je zgrajen na jedru operacijskega sistema Linux. Uporablja se za pametne telefone in tablične računalnike. Za razvoj je najbolj zaslužno podjetje Google, ki je tudi ustanovitelj poslovnega združenja Open Handset Alliance, ki si prizadeva za skupen razvoj odprtih standardov na področju mobilnih naprav. Po podatkih[10] podjetja IDC, ki se ukvarja z raziskavami trga, se je tržni delež pametnih telefonov z operacijskim sistemom Android v drugem četrtletju leta 2012 povečal na 68,1% ali dobre 104 milijone prodanih naprav. Med najbolj pomembne izdelovalce naprav s tem operacijskim sistemom lahko štejemo podjetja Samsung, HTC, Motorola in Sony. Obstaja pa še cela vrsta manj znanih proizvajalcev, ki prav tako ponujajo dolge palete najrazličnejših naprav. Ker imajo proizvajalci in tudi ljubitelji teh naprav možnost prilagoditev osnovnega sistema velja Android OS za izjemno fragmentirano platformo za razvoj, še posebej v primerjavi z največjim konkurentom operacijskim sistemom iOS podjetja Apple.

Podjetje Google je izbralo programski jezik Java za glavni steber pri izdelavi operacijskega sitema Android. Poleg dejstva, se je bila večina operacijskega sistema, ki temelji na jedru Linux 2.6 napisana v Javi, se isti jezik uporablja tudi pri razvojnem orodju Android SDK, ki je podlaga za ustvarjanje različnih aplikacij za to platformo.

Trenutna različica je Android 4.1, izdelana aplikacija pa podpira vse naprave do različice 2.2 nazaj. Ker je Android SDK izpeljan iz programskega jezika Java se lahko programska knjižnjica z implementacijo protokola HLS uporabi brez potreb po dodatnih spremembah.

Od različice Android 3.0 je podpora protokolu HLS verzije 2 ter od Android 4.0 verzija protokola 3 že del osnovnega sistema[5]. Ker pa je trenutno še vedno 90% naprav v uporabi starejših verzij od Android 3.0[6]. je tako lastna implementacija protokola nujno potrebna. Z lastno podporo verzije Android 2.2 ali novejše se tako na primer podpre več kot 90% naprav[6], ki so trenutno v uporabi. Hkrati razvijalci dobijo tudi boljši nadzor in pregled nad delovanjem. V praksi se je privzeta podpora tudi na novejših verzijah sistema sistema izkazala kot samo delna in nezanesljiva ter je bila zelo odvisna od posameznih naprav oziroma proizvajalcev.

4.1.3 Eclipse

Za razvojno okolje je bil izbran Eclipse IDE, ki je razvit prav v programskem jeziku Java. Podpora različnim programskim jezikom in operacijkim sistemom je osnova za uporabo v vrsti različnih namenov. Največja prednost je razširljiv sistem vtičnikov, ki prinaša vse od razvijanja v programskih jezikih kot so Python, C in PHP pa vse do razvijalskih orodij za modeliranje, na primer za standarda UML in BPMN. Eclipse je brezplačen in izdan pod odprtokodno licenco Eclipse Public License.

Integrirano razvojno okolje Eclipse pa samo po sebi ne bi bilo v pomoč brez vtičnika Android Developer Tools. Skupek orodij vsebuje konstruktor za izdelavo uporabniških vmesnikov. Na voljo je tudi simulator naprav Android poljubnih karakteristik in gonilniki za testiranje na fizičnih napravah. Tako simulator kot fizične naprave se lahko uporabijo za testiranje, razhroščevanje in profiliranje delovanja aplikacij. Meri se lahko poraba pomnilnika, zasedenost CPE, grafični pregled porabe omrežne hitrosti in drugih analiz.

Za upravljanje različic izvorne kode se je uporabil sistem Subversion. Za lažje delo iz razvojnega okolja Eclipse je bil uporabljen vtičnik Subclipse, ki je pokril vse potrebe opisanih projektov.

Kot alternativo bi bilo možno uporabiti tudi drug IDE a je Eclipse skupaj z vtičnikom ADT najboljše orodje za razvoj aplikacij na platformi Android. Ena alternativa je IntelliJ IDEA, ki pa ne dosega takše podpore različnim orodjem iz Android SDK kot tudi podpore skupnosti razvijalcev. Zadnja možnost je pisanje izvorne kode v poljubnem tekstovnem urejevalniku in uporaba orodij iz Android SDK preko ukazne vrstice.

4.1.4 Ostali pripomočki

- Za testiranje omejene mrežne povezave smo uporabili orodje imenovano Network Link Conditiner, ki je del razvojnega paketa Xcode na operacijskem sistemu Mac OS X. Orodje omogoča omejitev pasovne širine, nastavitev deleža izgubljenih paketov ter mrežno zakasnitev. Vse pogoje lahko določamo za obe smeri povezave posebej. Nekaj možnosti kot sta 3G in EDGE omrežji je že prednastavlenih. Seveda pa je del testiranja potekal tudi na fizičnih napravah različnih proizvajlcev ter na različnih omrežjih.

- Media Stream Validator Tool je orodje ukazne vrstice za potrjevanje pravilnosti HLS medijskih tokov in strežnikov. Orodje simulira delovanje

odjemalca in preverja, če indeksne datoteke ustrezajo specifikaciji protokola. Izvede tudi več preverjanj, ki podajo oceno o zanesljivost in najde morebitne napake kot so nepravilna enkripcija, manjkajoči kosi, napačni metapodatki glede na video segmente, itd... Orodje tako služi za dodatno potrditev pravilnosti delovanja strežnika in skladnost s protokolom.

4.2 Arhitektura knjižnjice

Celotna knjižnjica je postavljena v samostojen Javin paket, lahko pa se iz tega izvozi .jar datoteka za zunanjo uporabo. S tem se navzven odpre le programski vmesik. Dokumentacija je samodejno zgenerirana z uporabo orodja javadoc, ki vključi vse vidne razrede, funkcije in parametre.

Knjižnica deluje tako, da za vhod prejme URL do predvajnaja v protokolu HLS, torej pot do m3u8 datoteke. Pri implementaciji protokola je prišlo do spoznanja, da platforma Android ne omogoča predvajanja z vhodnim tokom *InputStream*, omogoča pa predvajanje poljubnega URL naslova ali datoteke iz fizičnega pomnilnika. Zaradi te omejitve je bila knjižnjica načrtovana tako, da kot izhod pripravi neomejen podatkovni tok videa na lokalni zanki mrežnega vmesnika localhost na poljubnih vratih. Torej poljuben predvajalnik vzpostavi povezavo na naslovu http://localhost:1234 in prejema podatke v neprekinjeni obliki.

Programska knjižnjica ne vsebuje nekaterih delov specifikacije protokola kot je enkripcija. Tudi avtentifikacija uporabnikov, ki sicer ni del protokola, je urejena ločeno in ni del programske knjižnjice.

4.2.1 Delovni razredi

Programska knjižnjica je sestavljena iz naslednjih razredov:

- LocalhostServer: Glavni razred, ki omogoča delovanje in povezuje ostale razrede. Vsebuje tudi vse niti, ki so podrobno opisane v naslednjem razdelku.

- PlaylistParser: Razred namenjen razčlenjevanju datotek, ki se prenesejo s strežnika. Kot vhodni parameter dobi URL naslov do seznama predvajanja. Vsebuje logiko za prenos seznamov predvajanja ter upošteva HTTP statusne kode kot je preusmeritev 301. Iz prenešene m3u8 datoteke zgradi drevesno strukturo instanc objektov Stream ali Segment, odvisno od tega ali gre za datoteko z različnimi kvalitetnimi nivoji ali za seznam segmentov za predvajnaje.

- HlsListener: Abstraktni vmesnik, ki je namenjen prenosu povratnih informacij o predvajanju. Vsaka uporaba te programske knjižnjice zahteva tudi implementacijo tega vmesnika. S tem se zagotovi, da bodo informacije kot so prekinitev predvajanja zaradi napake ali zamenjava kvalitetnega nivoja dosegle naslovnika, torej glavni del programa. Integratorju pa ponuja možnost, da se pri različnih uporabah knjižnjice različno glavni del programa različno odziva na določene dogodke.

- M3U8 Playlist: Razred katerega instance predstavljajo posamezne sezname predvajanja. Namenjen je tako seznamom kvalitenih nivojev kot seznamom video segmentov. Poleg omenjenih seznamov drži podatke o trenutno predvajanem segmentu in izbranem kvalitetnem nivoju. Glede na vhodni parameter o trenutni hitrosti omrežne povezave povezave izbira najprimernejši kvalitetni nivo. Skrbi za osveževanje seznama predvajanja. Torej iz pomikajočega seznama izločuje segmente, ki so se že predvajali in na konec dodaje nove segmente ter skrbi za to da ne pride do podvajanja segmentov. Razred vključuje tudi nekaj metod, ki so v pomoč pri različnih seznamih predvajanja z relativnimi potmi do segmentov.

- Stream: Instanca tega razreda predstavlja posamezen kvalitetni nivo. Vsebuje podatke o pasovni širini, identifikacijsko številko in URL pot do seznamov predvajanja za to kvaliteto. Implementira tudi vmesnik *Comparable* ter s tem omogoča enostavno sortiranje seznama kvalitetnih nivojev po atributu pasovne širine.

- Segment: Posamezen segment medijskega toka podatkov. Hrani vse atribute posameznega segmenta kot so ime, dolžina, URL in številka v sekvenci, ki izhajajo iz specifikacije protokola.

- HlsDemoPlayer: Razred, ki je namenjen izključno testiranju knjižnjice. Deluje brez dejanskega predvajalnika video vsebin in ga je možno zagnati neposredno iz razvijalnega okolja. Delovanje izpisuje v konzolo in tako pomaga pri iskanju napak v implementaciji.

4.2.2 Niti

Da se osveževanje seznama predvajanja in prenašanje segmentov odvija neodvisno od dostave teh segmentov predvajalniku je bila potrebna uporaba niti,

ki delujejo neodvisno druga od druge a vključujejo način za medsebojno komunikacijo ob dogodkih. Tako po začetni inicializaciji predvajanja tečeta dve niti:

- ServingThread: Glavna nit v neskončni zanki skrbi za osveževanje seznama predvajanja in prenos novih segmentov, ko se pojavijo. Ko prenese en segment iz seznama, sproži algoritem za ugotavljanje zamenjave kvalitetnega nivoja. Nit se za določen čas zaustavi v dveh primerih. Če je v določenem času prenesla za več kot eno minuto video segmentov počaka za toliko sekund, da se medpomnilink zmanjša na vrednost pod omenjeno mejo. Drug razlog za čakanje je ko nit osveži seznam predvajanja v katerem pa ni novih segmentov. Takrat počaka polovico maksimalnega časa enega segmenta, kot je pravilo v specifikaciji protokola. Do prvega primera čakanja pride torej v primeru videa na zahtevo, kjer so že na začetku znani vsi segmenti a ni smotrno prenašati več kot za 1 minuto videa vnaprej. Drugi primer pa je vezan izključno na predvajanje v živo, ko je potrebno najti pravo mejo med konstantnim osveževanjem seznama predvajanja in množici zahtev na strežnik ter prepočasnim osveževanjem seznama predvajanja, kjer lahko pride do tega, da zmanjka segmentov v vrsti za prenašanje.

- PushThread: Ta nit hrani verižni seznam prenešenih segmentov, ki še niso bili dostavljeni predvajalniku. V neskončni zanki iz verižnega seznama jemlje prvi kos in ga pošlje predvajalniku. Če ji v seznamu zmanjka video segmentov se ustavi za določen čas. Od zunaj ima poleg vmesnika za zagon in ustavitev še metodo za dodajanje segmenta v seznam, ki v primeru da je bil seznam prej prazen tudi zbudi nit iz spanja.

4.2.3 Podpora različnim aplikacijam

Kot je bilo že omenjeno v uvodu poglavja se knjižnjica lahko uporabi kjerkoli, kjer je na voljo programski jezik Java. Ker knjižnjica uporablja le elemente, ki so standardi večini objektno orientiranih programskih jezikom se lahko brez večjih težav celotno knjižnjico prepiše v drug objektno orientiran programski jezik.

4.2.4 Algoritem za menjavo pasovne širine

Pri vsakem prenosu novega segmenta se izračuna čas, ki je bil potreben za prenos. Glede na dolžino segmenta se izračuna hitrost s katero je bil segment prenešen s strežnika. Algoritem glede na ta podatek izbere najvišji kvalitetni nivo, ki zagotavlja prenos segmenta v 80% časa predvajanja tega segmenta. Takšen algoritem se dostikrat pojavlja tudi pri drugih predvajalnikih, včasih pa ti algoritmi računajo glede na hitrost prenosa zadnjih dveh segmentov da bolje preprečijo morebitno stalno preklapljanje med višjo in nižjo kvaliteto ko je hitrost povezave ravno nekje na sredi.

Ob začetku predvajanja algoritem izbere prvi kvalitetni nivo v seznamu. Tako dela tudi predvajalnik na Apple iOS. Takšna implementacija daje možnost, se lahko zunaj te logike določi začetni kvalitetni nivo. Navadno je to eden od slabših kvalitetnih nivojev, da se zagotovi hiter začetek predvajanja. Druga možnost bi bila pomnenje najbolj pogoste pasovne širine pri kateri se predvajanje ustali po začetnih menjavah kvalitet a bi s tem največkrat izbrali višji kvalitetni nivo in podaljšali čas do začetka predvajanja.

V primeru, da pride do menjave kvalitete algoritem ne zavrže že prenešenih segmenov, ki še niso bili predvajani. Zavrže pa vse še ne prenešene segmente iz prejšneja kvalitetnega nivoja in izbere le segmente iz novega.

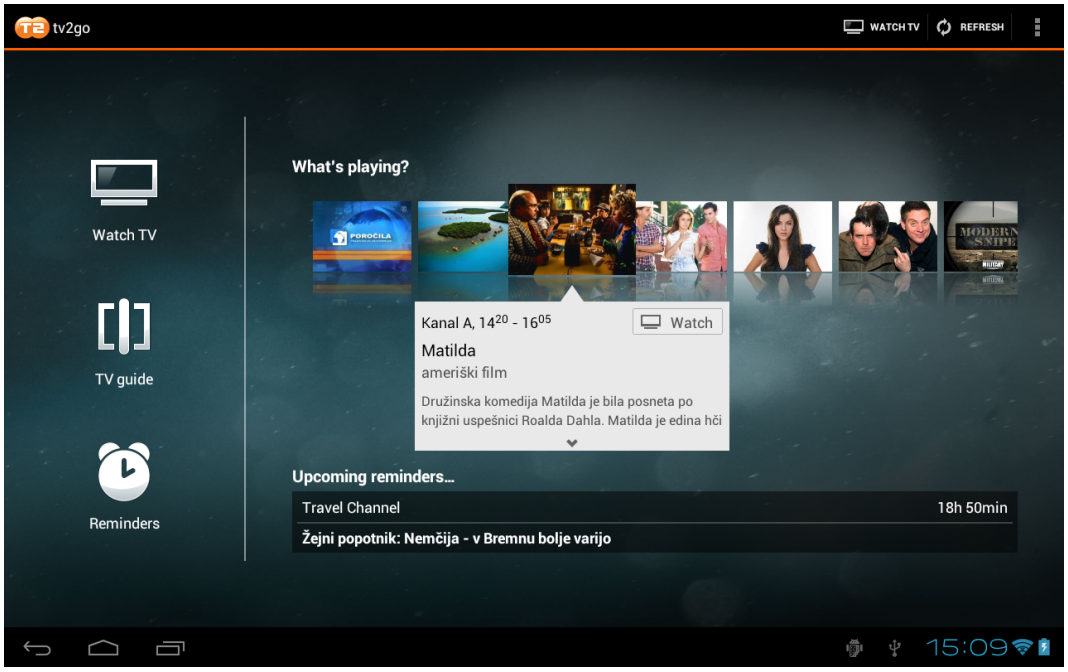
Obstajajo tudi primeri, ko ne želimo predvajanja s spremenljivo pasovno širino. V klicu na začetku predvajanja je na voljo parameter, ki omogoča preprečitev menjavanja kvalitetnih nivojev. Predvajalnik ves čas predvaja kvalitetni nivo na pri katerem je začel predvajanje.

Poleg ugotavljanja primerne pasovne širine predvajanja glede na hitrost povezave poznamo še omejevanje pasovne širine glede na strojne karakteristike napreve. Ta algoritem ni del te knjižnjice saj je odvisen od naprav na katerih teče. V primeru ko gre za osebne računalnike ta mehanizem v večini primerov ni potreben.

4.3 Predstavitev aplikacij in analiza delovanja knjižnjice

4.3.1 T-2 tv2go

Aplikacija T-2 tv2go omogoča spremljanje televizije na mobilnih napravah z operacijskim sistemom Android. Poleg tega omogoča brskanje po sporedih, nastavljanje opomnikov in uporabo drugih funkcij kot je prikazano na spodjnih



Slika 4.1: Začetni zaslon aplikacije T-2 tv2go

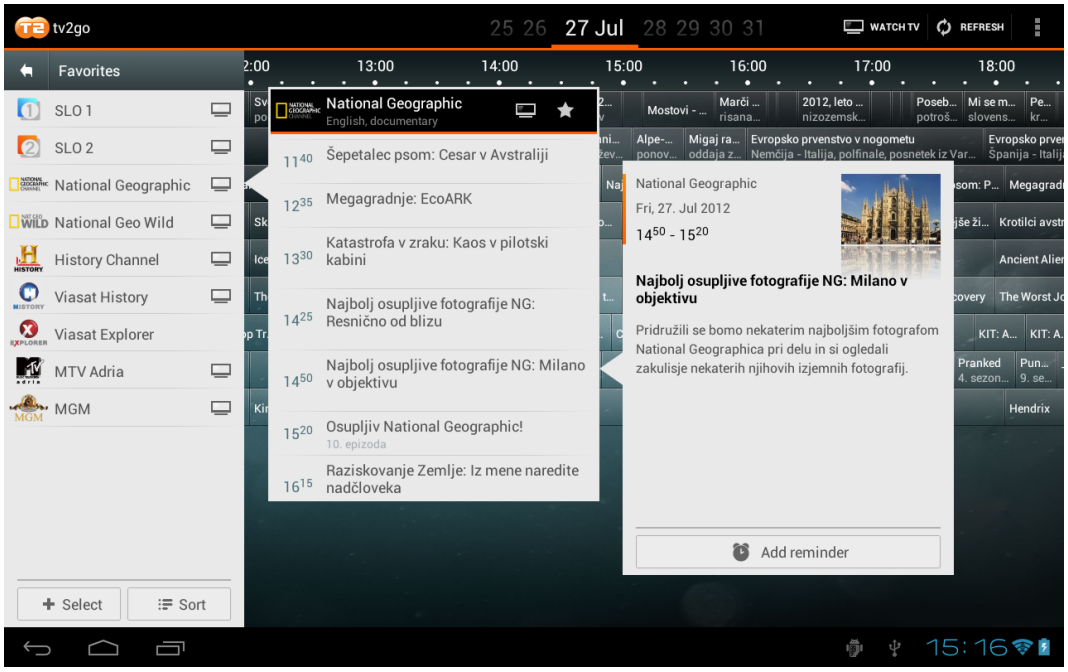
zaslonskih posnetkih. Spremljanje tv programov je na voljo v živo in za 48 ur nazaj.

Naša implementacija protokola je bila uporabljena v začetnih fazah razvoja in pri prvih prototipih. Delovanje je bilo solidno. Zaradi omejitev dekoderjev videa na določenih napravah, ki niso znali pravilno dekodirati zaporedja segmentov iz različnih kvalitetnih novojev, se je uporabil drug predvajalnik, ki je že vključeval podporo protokolu HLS. Še vedno pa je v aplikaciji vključen del kode, ki skrbi za povezljivost s strežnikom.

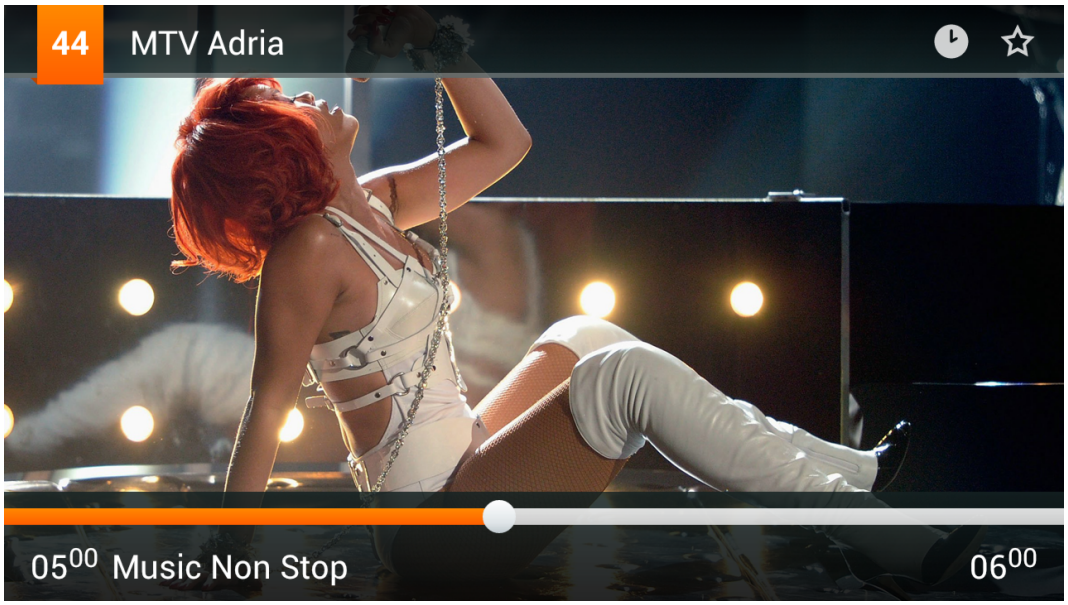
Poleg te aplikacije je na voljo podobna aplikacija za sistem iOS, ki pa uporablja sistemski predvajalnik, ki že vsebuje podporo protokolu HLS.

4.3.2 T-2 Agent

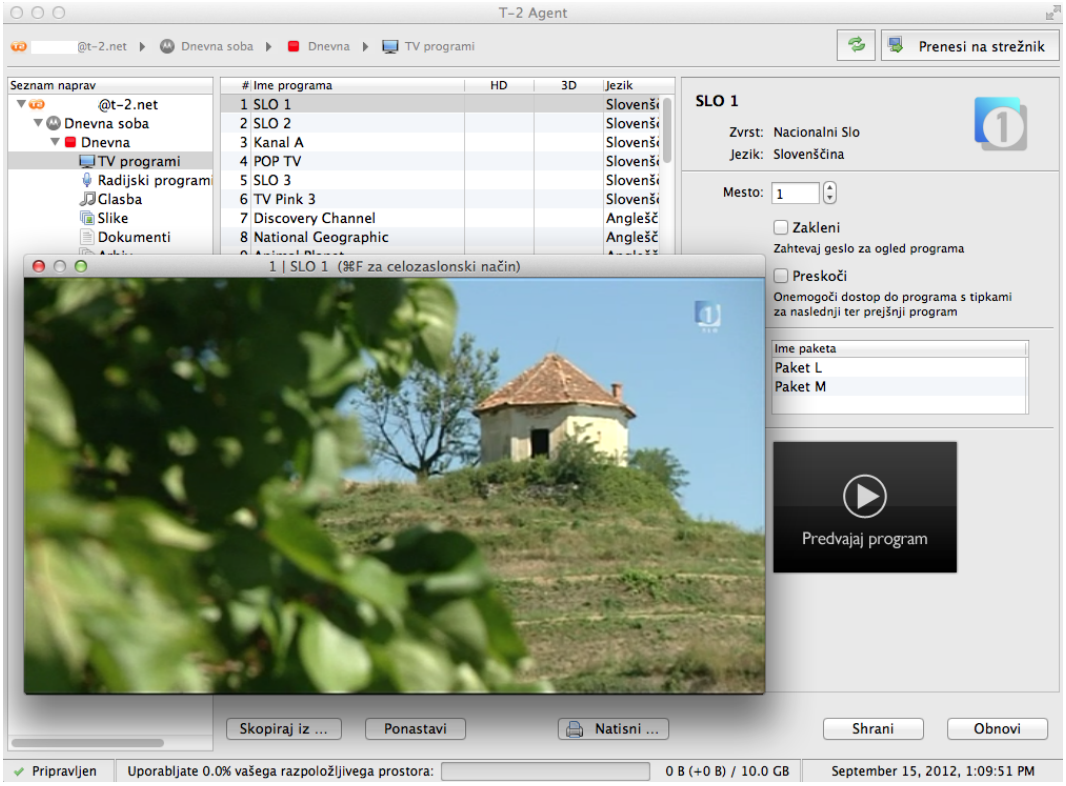
T-2 Agent je aplikacija za osebne računalnike, ki deluje na operacijskih sistemih Windows, OS X in Linux. Omogoča predvajanje TV programov v živo in še množico drugih funkcij za urejanje svoje glasbe, slikovnih albumov in dokumentov. Omogoča tudi urejanje vrstnega reda programov izbranega profila uporabnika iz nabora radijskih in televizijskih programov.



Slika 4.2: Pregled sporeda v aplikaciji T-2 tv2go



Slika 4.3: TV predvajalnik v aplikaciji T-2 tv2go



Slika 4.4: T-2 Agent pri predvajanju televizije na operacijskem sistemu OS X

Za predvajanje TV programov je bila uporabljena programska knjižnjica iz tega poglavja. Ker je aplikacija napisana v programskem jeziku Java in uporablja standarne programske funkcije ni bilo potrebe po večjih spremembah razen manjših dodelav za povečanje stabilnosti kar pa je omenjeno v naslednjem podpoglavju.

Na sliki zgoraj lahko vidimo zaslonski posnetek aplikacije med predvanjanjem TV programa. V ozadju je okno za nastavljanje televizijskih programov.

4.3.3 Analiza delovanja

Implementacija protokola deluje dobro tudi v produkcijskem okolju. Še vedno so odprte možnosti za implementacijo posameznih funkcij iz novejših različic specifikacije protokola, če bi se pojavila potreba po tem.

Posamezne dodelave so bile potrebene zaradi namenske uporabe predvajanja TV programov, pri katerem uporabniki pogosto menjajo med programi

in zahtevajo ogromno zaporednih zaustavitev in ponovnih vzpostavitev programa. Potrebno je bilo uvesti dodatne mehanizme, ki preverjajo, če novo predvajanja zapre vse prejšne niti in počisti vse objekte v pomnilniku, da ne pride do neljubih dogodkov.

Pri testiranju pri omejenih omrežnih povezavah se sicer preprost algoritev izkazal za dovolj dobrega. Ne glede na algoritem pa ne moremo preprečiti zaustavitve predvajanja ob hitrosti povezave, ki je nižja od najnižjega kvalitetnega nivoja. Prav tako se predvajanje prekine ob prekinitvi omrežne povezave za več kot je čas medpomnenja, ki je ponavadi 30 sekund.

Konkurenčni operacijski sitem iOS v svoji implementaciji omogoča preklope med različnimi kvalitetnimi nivoji tudi sredi segmentov. To precej izboljša uporabniško izkušnjo pri dobrih omrežnih povezavah, ko za preklop na višji kvalitetni novo ni potrebno čakati 10 sekund do naslednjega segmenta ampak se ta zgodi takoj ko je prenešen segment v novi kvaliteti. Žal zaradi omejitev v programskem vmesniku Android in kvalitete sistemskih dekoderjev na napravah tega ni bilo mogoče izvesti.

Ker poleg tehničnih obstajajo tudi druge vrste omejitev, je uporaba opisane implementacije odvisna tudi od zahtev lastnikov avtorskih pravic, ki lahko zahtevajo uporabo certificiranih predvajalnikov tujih ponudnikov.

Poglavje 5

Sklep

5.1 Kaj je narejeno, glavni prispevki

Glavna tema diplomske naloge je protokol Http Live Streaming. Ker je protokol šele v fazi osnutka in je samo eden izmed množice različnih je bilo smotrno na začetku naloge raziskati in opisati različne vrste protokolov in njihovo delovanje. V osrednjem delu se je na podlagi specifikacije in praktičnih izkušenj podrobno opisal protokol HLS. Glavna konkurenta v tem času MSS in HDS se na videz od HLS ne razlikujeta a je pomembno poznati bistvene razlike v delovanju. Implementacija protokola v programskem jeziku Java je prinesla knjižnjico, ki je neodvisna od platforme in operacijskega sistema in se lahko v celoti ali delno uporabi pri naslednjih podobnih projektih. Čeprav se implementacija protokola glede na prvotne načrte ni v celoti uporabila v aplikaciji za Android, je razvoj prinesel mnogo spoznanj in nam približal razumevanje delovanja in pohitril odpravo napak, ki se pojavljajo v podobnih okoliščinah.

Diplomsko delo je pomembno tudi iz stališča dokumentacije s tega področja v slovenščini, saj je do tega trenutka edino delo v slovenščini, ki opisuje ta protokol in je lahko marsikomu dobra osnova za morebitne implementacije v drugih programskih jezikih.

5.2 Uporaba

HLS knjižnjica se v času pisanja diplome uporablja v aplikaciji T-2 Agent za osebne računalnike z operacijskimi sistemi Windows, Mac OS X ter Linux. Knjižnjica je bila primarno razvita za aplikacijo za naprave z operacijskim

sistemom Android in je bila tudi vključena v testnih verzijah aplikacije a se je zaradi zahtev lastnikov avtorskih pravic ter problemov zaradi razdrobljenosti platforme uporabil drug predvajalnik, ki podporo za HLS protokol že vključuje tako da vključitev v končno aplikacijo ni bila potrebna.

5.3 Možne dopolnitve in izboljšave

Protokol ni bil implementiran v celoti. Možno bi bilo dodati del, ki se tiče enkripcije posameznih segmentov ter ostalih možnih oznak, ki se lahko pojavljajo v seznamih predvajanja. Vsi ti so iz najnovejše različice HLS protokola. Ob dobrem poznavanju ne bi bilo pretežko implementirati protokol HLS tudi v druge programske jezike oziroma platforme, kjer protokol še ni podprt (npr. Windows Phone 7). Za operacijski sistem Android pa bi lahko pripravili SDK, ki bi bil po funkcijah podoben tistemu za iOS.

Velik iziv pri pisanju diplomske naloge je bilo iskanje ustreznih slovenskih besed. Še posebej za novejše protokole in tehnologije je bilo težko najti ustrezne izraze v slovenščini, ki bi jasno razločevali med posameznimi pojmi. V veliko pomoč je bil terminološki slovar informatike islovar[12] v katerem pa še vedno manjka precej ustreznih besed. Tako je lahko izboljšava tudi predlog dopolnitve slovarja in spodbuda strokovnjakom, da se najde ustrezne izraze in se jih tudi uporablja v strokovnih člankih in knjigah v prihodnosti.

Dodatek A

Primeri seznamov predvajanja

A.1 Seznam kvalitetnih nivojev

#EXTM3U
#EXT-X-STREAM-INF:PROGRAM-ID=1,BANDWIDTH=1280000
http://example.com/low.m3u8
#EXT-X-STREAM-INF:PROGRAM-ID=1,BANDWIDTH=2560000
http://example.com/mid.m3u8
#EXT-X-STREAM-INF:PROGRAM-ID=1,BANDWIDTH=7680000
http://example.com/hi.m3u8
#EXT-X-STREAM-INF:PROGRAM-ID=1,BANDWIDTH=65000,CODECS="mp4a"
http://example.com/audio-only.m3u8

A.2 Pomikajoče okno medijskih segmentov

#EXTM3U
#EXT-X-TARGETDURATION:9
#EXT-X-MEDIA-SEQUENCE:56169
#EXTINF:8,
http://liveips.nasa.gov.edgesuite.net/msfc/Wifi/Wifi_052912_074957_56169.ts
#EXTINF:8,
http://liveips.nasa.gov.edgesuite.net/msfc/Wifi/Wifi_052912_074957_56170.ts
#EXTINF:8,
http://liveips.nasa.gov.edgesuite.net/msfc/Wifi/Wifi_052912_074957_56171.ts
#EXTINF:8,

http://liveips.nasa.gov.edgesuite.net/msfc/Wifi/Wifi_052912_074957_56172.ts
#EXTINF:8,
http://liveips.nasa.gov.edgesuite.net/msfc/Wifi/Wifi_052912_074957_56173.ts
#EXTINF:8,
http://liveips.nasa.gov.edgesuite.net/msfc/Wifi/Wifi_052912_074957_56174.ts
#EXTINF:8,
http://liveips.nasa.gov.edgesuite.net/msfc/Wifi/Wifi_052912_074957_56175.ts
#EXTINF:8,
http://liveips.nasa.gov.edgesuite.net/msfc/Wifi/Wifi_052912_074957_56176.ts
#EXTINF:8,
http://liveips.nasa.gov.edgesuite.net/msfc/Wifi/Wifi_052912_074957_56177.ts
#EXTINF:8,
http://liveips.nasa.gov.edgesuite.net/msfc/Wifi/Wifi_052912_074957_56178.ts
#EXTINF:8,
http://liveips.nasa.gov.edgesuite.net/msfc/Wifi/Wifi_052912_074957_56179.ts

Slike

Tabele

Literatura

[1] Saamer Akhshabi, Ali C Begen, Constantine Dovrolis, An experimental evaluation of rate-adaptation algorithms in adaptive streaming over http, MMSys', 2011

[2] Apple Inc., HTTP Live Streaming Overview, 2011, Dostopno na: http://developer.apple.com/library/ios/documentation/NetworkingInternet/Conceptual/StreamingMediaGuide/StreamingMediaGuide.pdf

[3] R. Pantos, Ed., HTTP Live Streaming, 2012, Dostopno na: http://tools.ietf.org/html/draft-pantos-http-live-streaming-09

[4] R. Pantos, HTTP Live Streaming Update, 2011, Dostopno na: http://adcdownload.apple.com/wwdc_2011/adc_on_itunes__wwdc11_sessions__pdf/408_http_live_streaming_update.pdf

[5] Android Developers, Android Supported Media Formats, 2012, Dostopno na: http://developer.android.com/guide/appendix/media-formats.html

[6] Android Developers, Platform Versions, 2012, Dostopno na: http://developer.android.com/resources/dashboard/platform-versions.html

[7] The Internet Engineering Task Force, 2012, Dostopno na: http://www.ietf.org/

[8] YouTube statistics, 2012, Dostopno na: http://www.youtube.com/t/press_statistics/

[9] Market Share: Mobile Devices by Region and Country, 4Q11 and 2011, 2012, Dostopno na: http://www.gartner.com/it/page.jsp?id=1924314

[10] IDC, Android and iOS Surge to New Smartphone OS Record in Second Quarter, Dostopno na:
http://www.idc.com/getdoc.jsp?containerId=prUS23638712

[11] Jan Ozer, Streaming Gets Smarter: Evaluating the Adaptive Streaming Technologies, 2009, Dostopno na:
http://www.streamingmedia.com/Articles/ReadArticle.aspx?ArticleID=65544

[12] islovar, Dostopno na:
http://www.islovar.org/